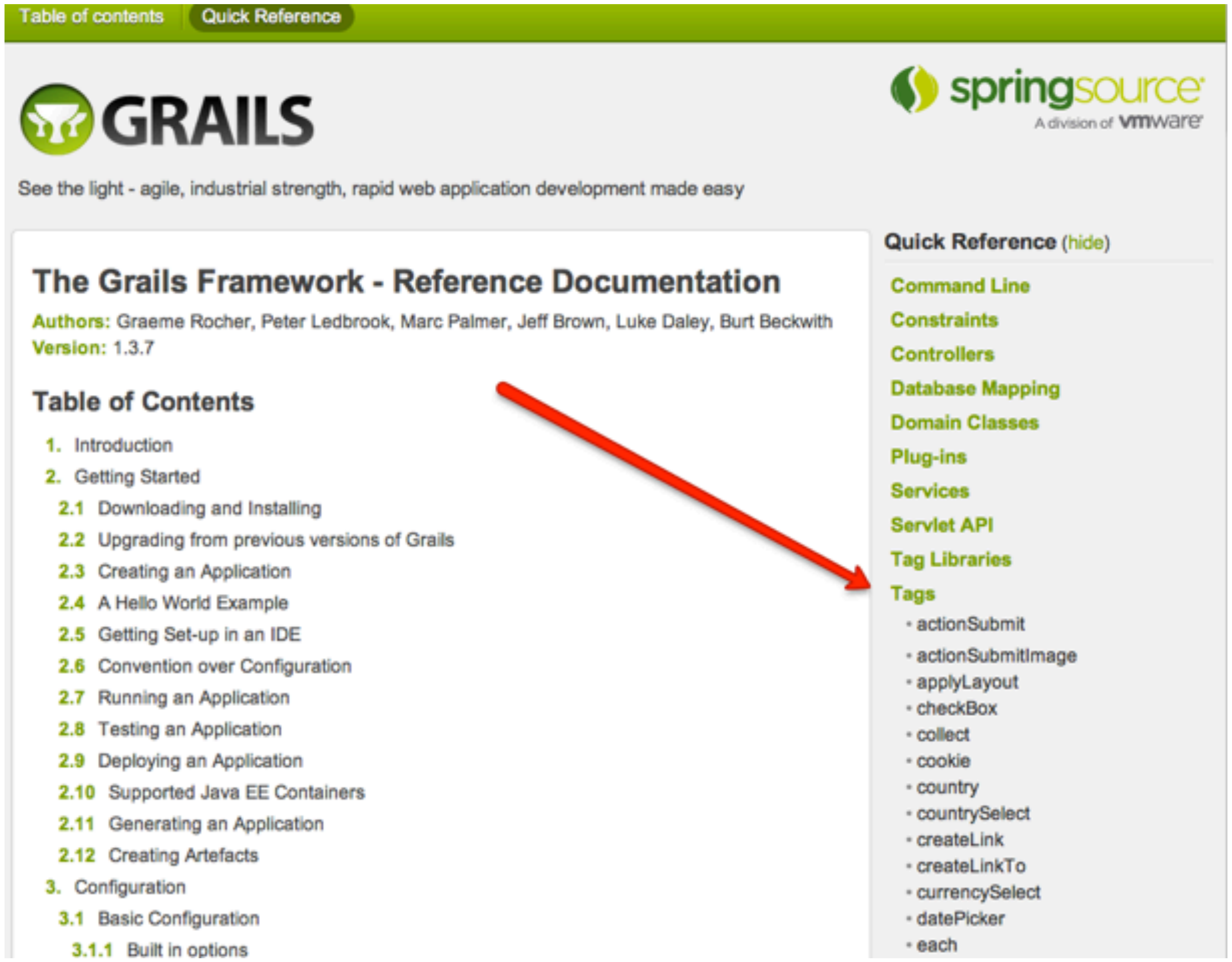
Tag Libraries

Tag Libraries

- Encapsulate display/rendering logic into reusable components
- Keep your views simple, more HTML-like
- Easier to perform automated testing on a tag than logic directly in a GSP

Built In Tag Libraries

- Extensive set of tags built in to Grails



See the light - agile, industrial strength, rapid web application development made easy

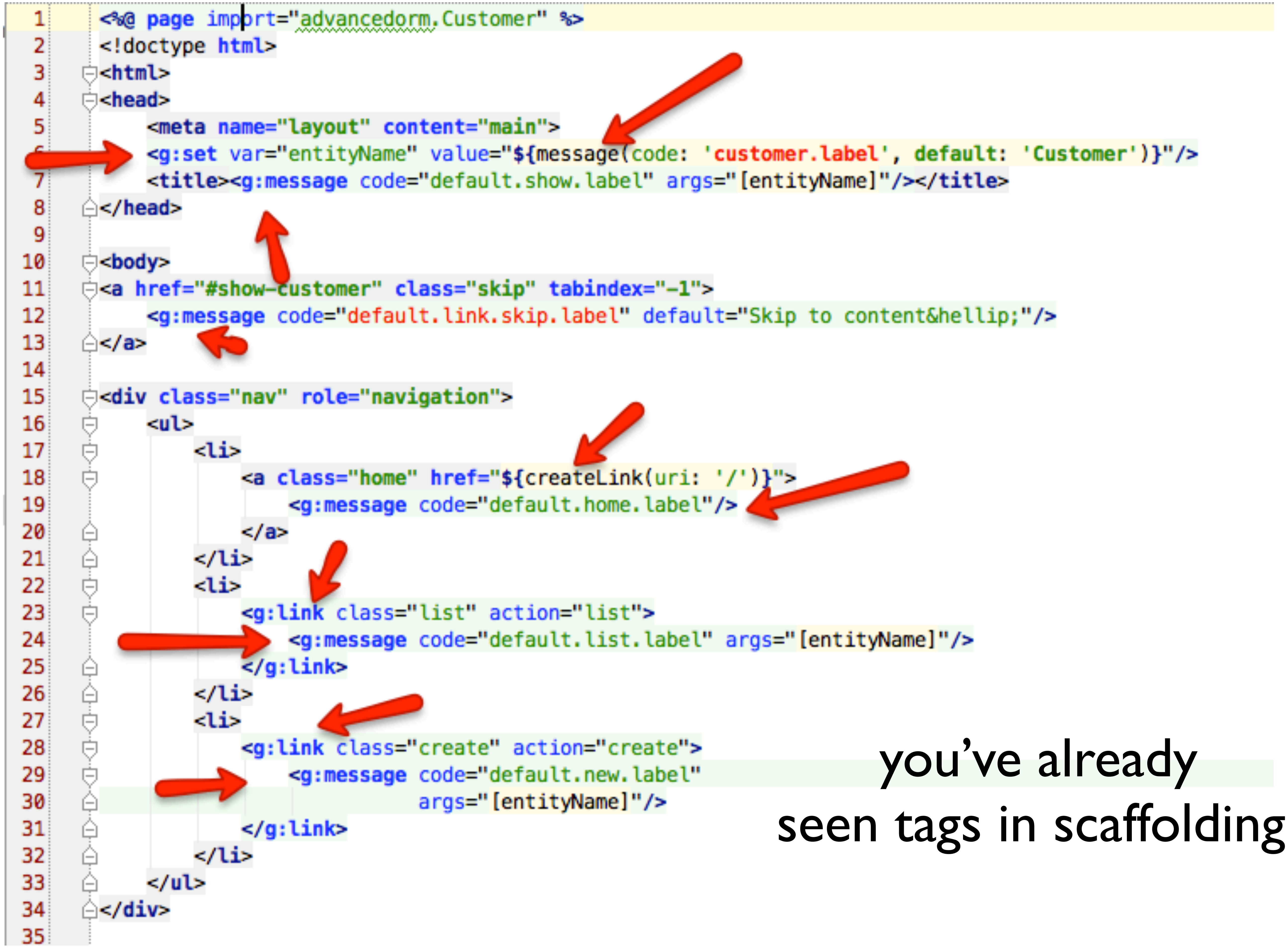
The Grails Framework - Reference Documentation

Authors: Graeme Rocher, Peter Ledbrook, Marc Palmer, Jeff Brown, Luke Daley, Burt Beckwith
Version: 1.3.7

Table of Contents

Quick Reference (hide)

- Command Line
- Constraints
- Controllers
- Database Mapping
- Domain Classes
- Plug-ins
- Services
- Servlet API
- Tag Libraries
- Tags
  - actionSubmit
  - actionSubmitImage
  - applyLayout
  - checkBox
  - collect
  - cookie
  - country
  - countrySelect
  - createLink
  - createLinkTo
  - currencySelect
  - datePicker
  - each



```
<%@ page import="advancedorm.Customer" %>
<!doctype html>
<html>
<head>
	<meta name="layout" content="main">
	<g:set var="entityName" value="${message(code: 'customer.label', default: 'Customer')}"/>
	<title><g:message code="default.show.label" args="[entityName]"/></title>
</head>

<body>
<a href="#show-customer" class="skip" tabindex="-1">
	<g:message code="default.link.skip.label" default="Skip to content&hellip;"/>
</a>

<div class="nav" role="navigation">
	<ul>
		<li>
			<a class="home" href="${createLink(uri: '/')}">
				<g:message code="default.home.label"/>
			</a>
		</li>
		<li>
			<g:link class="list" action="list">
				<g:message code="default.list.label" args="[entityName]"/>
			</g:link>
		</li>
		<li>
			<g:link class="create" action="create">
				<g:message code="default.new.label"
						   args="[entityName]"/>
			</g:link>
		</li>
	</ul>
</div>

```

you’ve already seen tags in scaffolding!

Logical

- if/else

```
<g:if test="${session.role == 'admin'}">
   <%-- show administrative functions --%>
</g:if>
<g:else>
   <%-- show basic functions --%>
</g:else>
```

Logical

- if/else - environment

```
<g:if env="development">
     Dev mode - debug: $someDebug
</g:if>
```

Logical

- unless
  - does the opposite of what the "if" tag would have done with the same condition

```
<g:unless test="${name == 'fred'}">
     Hello ${name}!
</g:unless>
```

Iterative

- each

```
<g:each in="${books}">
     <p>Title: ${it.title}</p>
     <p>Author: ${it.author}</p>
</g:each>

<g:each in="${books}" var="book">
     <p>Title: ${book.title}</p>
     <p>Author: ${book.author}</p>
</g:each>
```

Iterative

- while

```
<g:while test="${i < 5}">
    <%i++%>
    <p>Current i = ${i}</p>
</g:while>
```

Links

```
<g:link action="show" id="1">Book 1</g:link>
// results in <a href="/book/show/1">Book 1</a>


<g:link action="show" id="${currentBook.id}">
  ${currentBook.name}
</g:link>
// given currentBook.name = 'A Book' and
// given currentBook.id = 2
// results in <a href="/book/show/2">A Book</a>


<g:link controller="book">Book Home</g:link>
// results in <a href="/book/index">Book Home</a>


<g:link controller="book" action="list">Book List</g:link>
// results in <a href="/book/list">Book list</a>
```

Links

```
<g:link url="[action:'list',controller:'book']">
    Book List
</g:link>
// results in <a href="/book/list">Book List</a>


<g:link action="list"
  params="[sort:'title',order:'asc']">
     Book List
</g:link>
// results in
// <a href="/book/list?sort=title&order=asc">Book List</a>
```

Forms & Fields

- form

```
<g:form name="myForm" action="myaction" id="1">...</g:form>
//results in:
<form action="/shop/book/myaction/1" method="post"
   name="myForm" id="myForm" >...</form>

<g:form name="myForm" url="[action:'list',controller:'book']">..
</g:form>
//results in:
<form action="/shop/book/list" method="post" name="myForm"
id="myForm" >...</form>

<g:form action="show">...</g:form>
//results in:
<form action="/shop/book/show" method="post" >...</form>
```

Forms & Fields

- textField

```
<g:textField name="myField" value="${myValue}" />

// given myValue = 'abc123'
// results in
<input type="text" name="myField" value="abc123"/>
```

Forms & Fields

- checkBox

```
<g:checkBox name="myCheckbox" value="${true}" />

// results in
<input type="checkbox" name="myCheckbox"
checked="checked">
```

Forms & Fields

- hiddenField

```
<g:hiddenField name="myField" value="myValue" />

// results in
<input type="hidden" name="myField"
value="myValue">
```

Forms & Fields

- select

```
// create select from a list of books
// note the 'optionKey' is set to the id of each book element
<g:select name="store.book.id"
          from="${Book.list()}"
          value="${store.book.id}"
          optionKey="id"
          optionValue="title"/>


// given a list with one book (1: The First Book)
// results in
<select name="store.book.id">
<option value="1">The First Book</option>
</select>
```

Resources Tag

```
<g:resource dir="css" file="main.css" />
//results in /shop/css/main.css

<g:resource dir="css" file="main.css"
   absolute="true"/>
//results in
   http://portal.mygreatsite.com/css/main.css

<g:resource dir="css" file="main.css"
   base="http://admin.mygreatsite.com"/>
//results in
   http://admin.mygreatsite.com/css/main.css
```

Invoking Tags

- Tags can be invoked in controllers as method calls
- Controller:

```
g.link (action:"show" id:"1"){"Book 1"}
```

is the same as

- View:

```
<g:link action="show" id="1">Book 1</g:link>
```

Writing your own Tags

- Java Developers - have you ever written your own tag library?
- grails create-tag-lib
- Custom tags go in grails-app/taglib
- define closures with “attributes” (and optional body) parameter
- Can be namespaced using
  - static namespace = 'custom'
  - then referenced using that namespace:
  - <custom.myTag ...>

Testing

- Class Under Test == TagLibrary

```
@TestFor(SimpleTagLib)
class SimpleTagLibTests {
}
```

- Class Under Test == Artefact Utilizing Tag Library

```
@TestFor(SimpleController)
@Mock(SimpleTagLib)
class SimpleControllerUtilizingTagLib {
}
```

Testing - Assertions

- standard asserts will work
- Two helper assertion methods provided for you:
  - assertOutputEquals ('Hello World', '<s:hello />')
  - assertOutputMatches (/.*Fred.*/, '<s:hello name="Fred" />')

Anatomy of a Tag Library

```
class ServiceLevelTagLib {

    static namespace = "sl"

    def renderWithColor = {attrs ->
        if (attrs.serviceLevel){
            out << """<span class="servicelevel..."""
        }
    }

}
```

```
can be invoked with:
  <sl:renderWithColor serviceLevel="${serviceLevel}"/>
```

namespace

```
class ServiceLevelTagLib {

    static namespace = "sl"

    def renderWithColor = {attrs ->
        if (attrs.serviceLevel){
            out << """<span class="servicelevel..."""
        }
    }

}
```

can be invoked with:

```
 <sl:renderWithColor serviceLevel="${serviceLevel}"/>
```

tag name

```
class ServiceLevelTagLib {

    static namespace = "sl"

    def renderWithColor = {attrs ->
        if (attrs.serviceLevel){
            out << """<span class="servicelevel..."""
        }
    }

}
```

```
can be invoked with:
  <sl:renderWithColor serviceLevel="${serviceLevel}"/>
```

attributes

```
class ServiceLevelTagLib {

    static namespace = "sl"

    def renderWithColor = {attrs ->
        if (attrs.serviceLevel){
            out << """<span class="servicelevel..."""
        }
    }

}
```

```
can be invoked with:
  <sl:renderWithColor serviceLevel="${serviceLevel}"/>
```

writing to output stream

```
class ServiceLevelTagLib {

    static namespace = "sl"

    def renderWithColor = {attrs ->
        if (attrs.serviceLevel){
            out << """<span class="servicelevel..."""
        }
    }

}
```

```
can be invoked with:
  <sl:renderWithColor serviceLevel="${serviceLevel}"/>
```

body attribute

```
class ServiceLevelTagLib {

    static namespace = "sl"

    def ifGold = {attrs, body ->
        if (attrs.serviceLevel.name == 'Gold'){
            out << body()
        }
    }

}
```

```
can be invoked with:
    <sl:ifGold serviceLevel="${serviceLevel}">
        <!-- body content goes here -->
    </sl:ifGold>
```