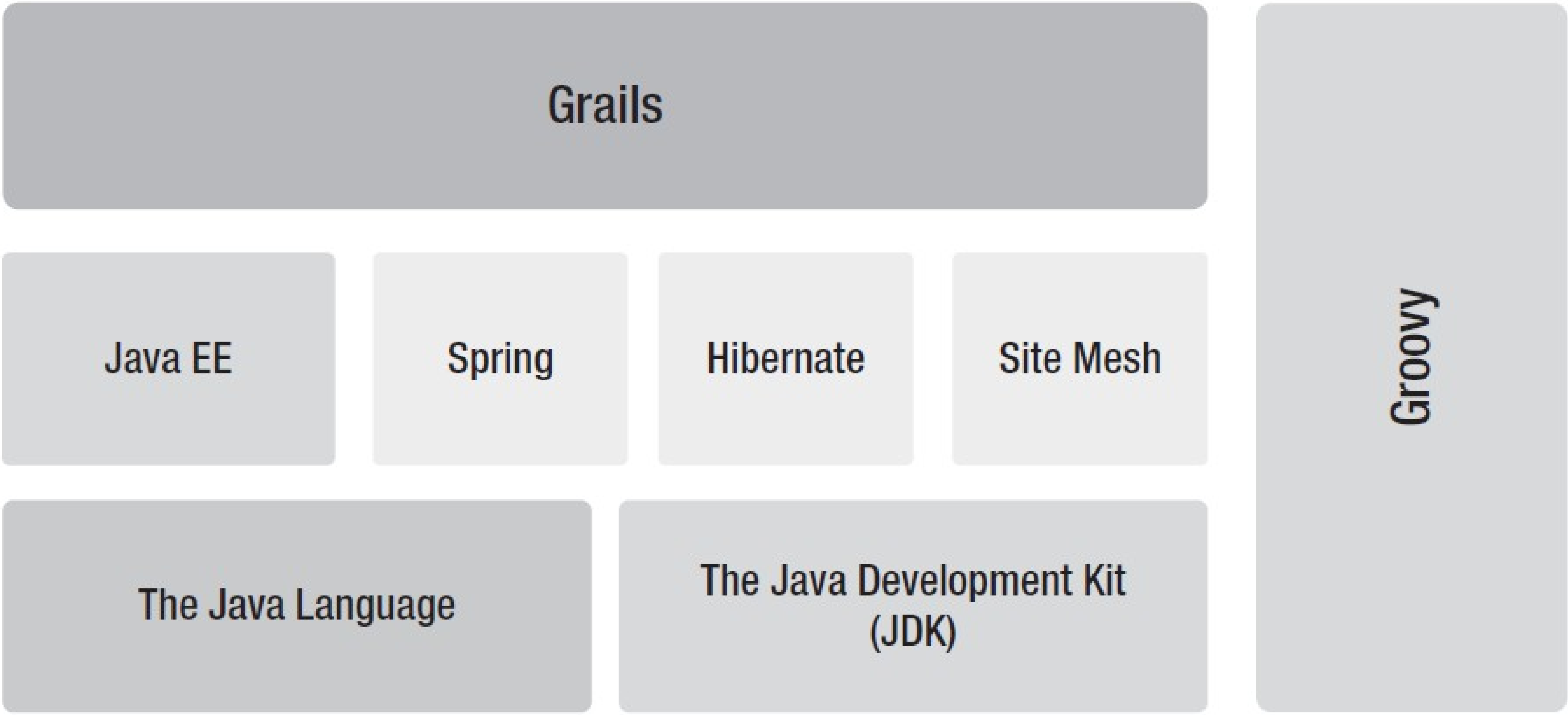
Grails


Grails
Java EE
Spring
Hibernate
Site Mesh
The Java Language
The Java Development Kit (JDK)
Groovy

Standard validators

Name	Example	Description
blank	login(blank:false)	Set to FALSE if a string value cannot be blank
creditCard	cardNumber(creditCard:true)	Set to TRUE if the value must be a credit-card number
email	homeEmail(email:true)	Set to 1 if the value must be an e-mail address
inList	login(inList:['Joe',	'Fred']) Value must be contained within the given list
min	duration(min:1)	Sets the minimum value
minSize	children(minSize:5)	Sets the minimum size of a collection or number property
matches	login(matches:/[a-zA-Z]/)	Matches the supplied regular expression
max	age(max:99)	Sets the maximum value
maxSize	children(maxSize:25)	Sets the maximum size of a collection or number property
notEqual	login(notEqual:'Bob')	Must not equal the specified value
nullable	age(nullable:false)	Set to FALSE if the property value cannot be null
range	age(range:16..59)	Set to a Groovy range of valid values
scale	salary(scale:2)	Set to the desired scale for floating-point numbers
size	children(size:5..15)	Uses a range to restrict the size of a collection or number
unique	login(unique:true)	Set to TRUE if the property must be unique
url	homePage(url:true)	Set to TRUE if a string value is a URL address

Domain class relationships

```
class Car {
   Engine engine
}
class Engine {
   Car car
}
```

```
class Car {
   Engine engine
}
class Engine {
   static belongsTo = [car:Car]
}
```

```
class Car {
   static hasOne = [engine: Engine]
}
```

```
class Engine {
   static belongsTo = Car
}
```

Domain class relationships

```
class Artist {
  String name
  static hasMany = [albums:Album]
}
class Album {
  String title
  static hasMany = [songs:Song]
  static belongsTo = [artist:Artist]
}
class Song {
  String title
  Integer duration
  static belongsTo = Album
}
```

```
class Album {
  String title
  static hasMany = [songs:Song]
  static belongsTo = [artist:Artist]
  SortedSet songs
}
```

```
class Album {
  String title
  static hasMany = [songs:Song]
  static belongsTo = [artist:Artist]
  static mapping = {
    songs cascade: 'delete'
  }
}
```

```
class Artist {
String name
  static hasMany = [albums:Album, instruments:Instrument]
}
```

Extending classes with inheritance

```
class Person {
  String firstName
  String lastName
  Integer age
}
class Employee extends Person {
  String employeeNumber
  String companyName
}
class Player extends Person {
  String teamName
}
```

```
class Employee extends Person {
  String employeeNumber
  String companyName
  static mapping = {
    // the value of the discriminator column for
    // Employee instances should be 'working people'
    discriminator 'working people'
  }
}
```

Extending classes with inheritance

Name	Description
value	The value to use for the discriminator
column	The name of the column for storing the discriminator
formula	An SQL expression that is executed to evaluate the type of the class
type	The Hibernate type

```
class Employee extends Person {
  String employeeNumber
  String companyName
  static mapping = {
    discriminator value: '42', type: 'integer'
  }
}
```

Extending classes with inheritance

```
class Person {
  String firstName
  String lastName
  Integer age
  static mapping = {
    tablePerHierarchy false
  }
}
```

Embedding objects

```
class Car {
  String make
  String model
  Engine engine
}
class Engine {
  String manufacturer
  Integer numberOfCylinders
}
```

```
class Car {
  String make
  String model
  Engine engine
  static embedded = ['engine']
}
```

Controllers - setting the default action

```
// Here the 'list' action is the default as there is only one action defined
class SampleController {
  def list() {}
}
// In this example 'index' is the default by convention
class SampleController {
  def list() {}
  def index() {}
}

// Here 'list' is explicitly set as the default
class SampleController {
  static defaultAction = 'list'
  def list() {}
  def index() {}
}
```

Logging

```
public interface Log {
  public void debug(Object msg);
  public void debug(Object msg, Throwable t);
  public void error(Object msg);
  public void error(Object msg, Throwable t);
  public void fatal(Object msg);
  public void fatal(Object msg, Throwable t);
  public void info(Object msg);
  public void info(Object msg, Throwable t);
```

```
  public void trace(Object msg);
  public void trace(Object msg, Throwable t);
  public void warn(Object msg);
  public void warn(Object msg, Throwable t);
  public boolean isDebugEnabled();
  public boolean isErrorEnabled();
  public boolean isFatalEnabled();
  public boolean isInfoEnabled();
  public boolean isTraceEnabled();
  public boolean isWarnEnabled();
}
```

Logging

```
class SampleController {
  def index() {
    log.info('In the index action...')
    // ...
  }
}
```

```
class SampleController {
  def index() {
    try {
      // do something that might throw an exception
    } catch (Exception e) {
      log.error ('some message goes here', e)
    }
  }
}
```

Accessing request attributes

Attribute	Description
actionName	The name of the currently executing action
actionUri	The relative URI of the executing action
controllerName	The name of the currently executing controller
controllerUri	The URI of executing controller
flash	The object for working with flash scope
log	An org.apache.commons.logging.Log instance
params	A map of request parameters
request	The HttpServletRequest object
response	The HttpServletResponse object
session	The HttpSession object
servletContext	The ServletContext object

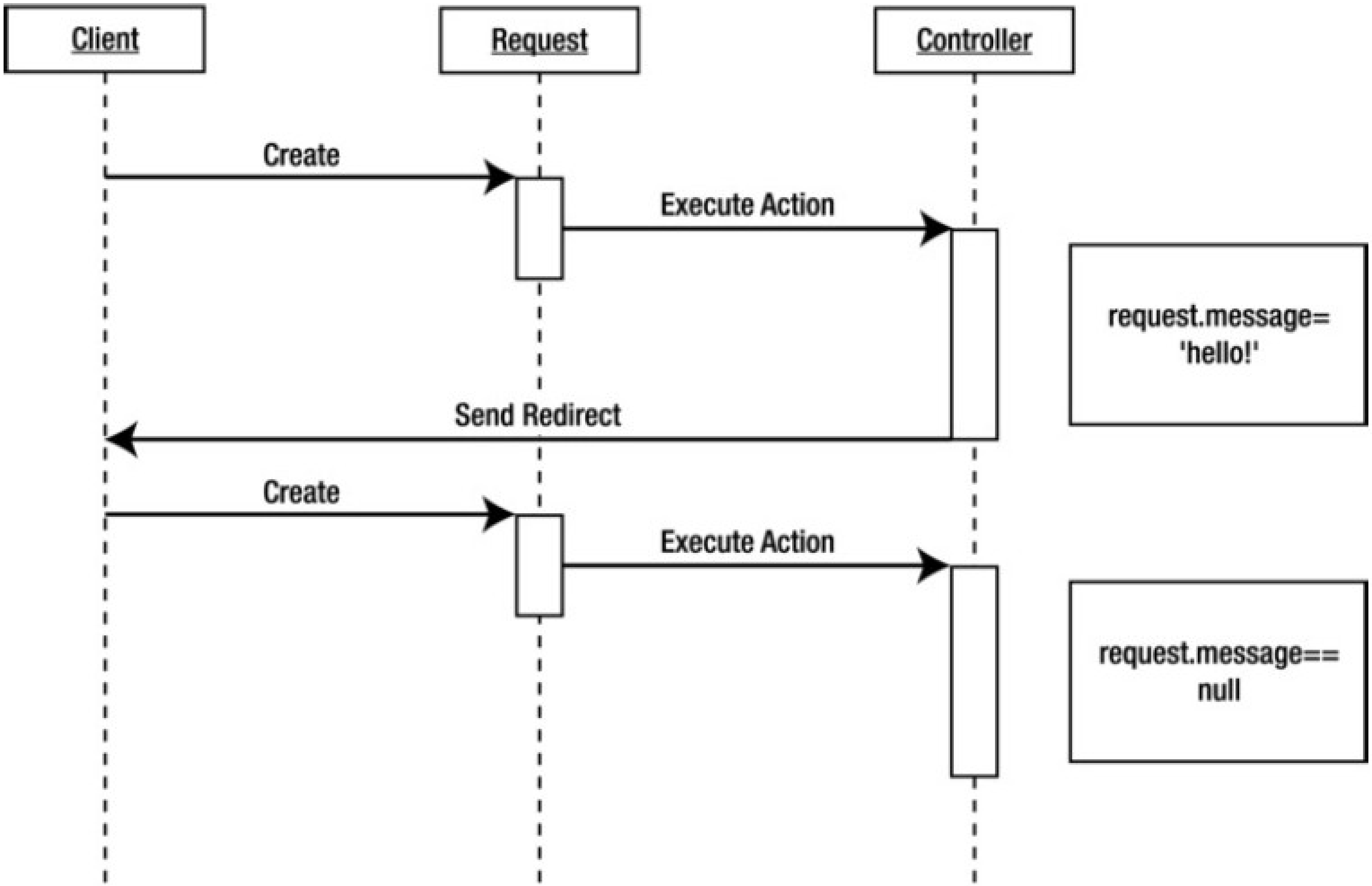
Accessing request attributes

Java Servlet	Grails Controller
request.getAttribute(“myAttr”);	request.myAttr
request.setAttribute(“myAttr”, “myValue”);	request.myAttr = “myValue”
session.getAttribute(“mAttr”);	session.myAttr
session.setAttribute(“myAttr”, “myValue””);	session.myAttr = “myValue”
servletContext.getAttribute(“mAttr”);	servletContext.myAttr
servletContext.setAttribute(“myAttr”, “myValue””);	servletContext.myAttr = “myValue”

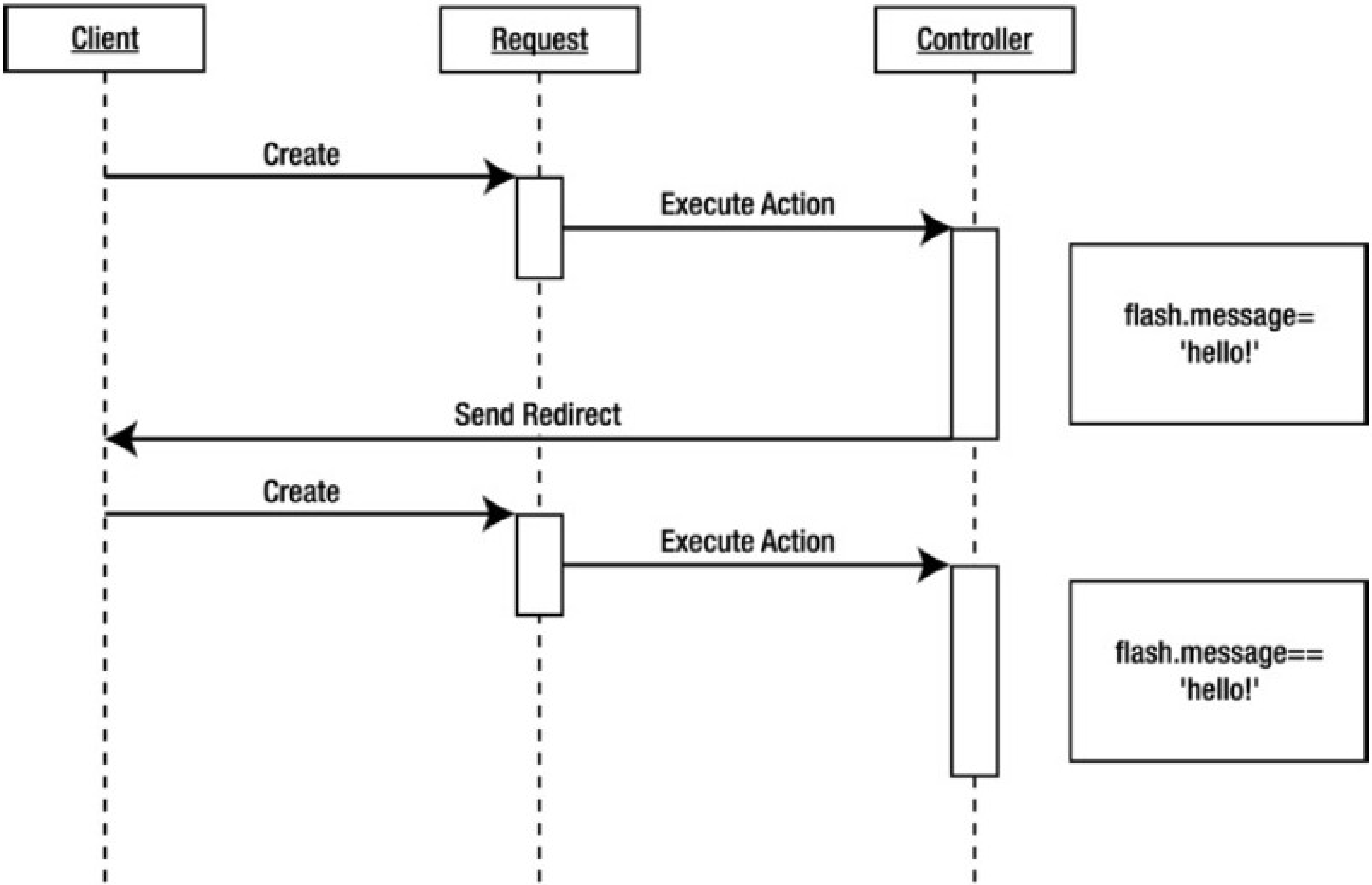
Using controller scopes

- The following list defines all the scopes available in order of their longevity
  - request: Objects placed into the request are kept for the duration of the currently executing request.
  - flash: Objects placed into flash are kept for the duration of the current request and the next request only.
  - session: Objects placed into the session are kept until the user session is invalidated, either manually or through expiration.
  - servletContext: Objects placed into the servletContext are shared across the entire application and kept for the lifetime of the application.

Message scope



Flash scope


Client
Request
Controller
Create
Execute Action
flash.message=
'hello!'
Send Redirect
Create
Execute Action
flash.message==
'hello!'

Accessing request parameters

```
def userName = request.getParameter('userName')
log.info("User Name: ${userName}")

def userName = params.userName
log.info("User Name: ${userName}")
```

```
def index() {
  // since params.counter is a string, it must be converted to an int if
  // the application intends to use the value as a number
  def counter = params.counter.toInteger()
  // …
}

def index() {
  def counter = params.int('counter')
  // …
}
```

Accessing request parameters

```
def index() {
for (name in params.list('name') {
  // do something with name
  }
}
```

```
def firstAction() {
  def counter = params.int('counter')
  def name = params.name
  // …
}
def secondAction(int counter, String name) {
  // there is no need to interact with the params object as
  // the request parameters have been bound to the counter
  // and name method arguments
}
```

Accessing request parameters

```
import grails.web.RequestParameter
class AdminController {
  // mainNumber will be initialized with the value
  // of params.accountNumber
  // accountType will be initialized with params.int('accountType')
  def action(@RequestParameter('accountNumber') String mainNumber,
             int accountType) {
      // ...
  }
}
```

Accessing request parameters

```
class AdminController {
  def action(String accountNumber, int accountType) {
    if(accountNumber == null && errors.hasErrors()) {
      def accountNumberError = errors.getFieldError(□accountNumber')
      if(accountNumberError != null) {
        // accountNumberError is an instance of
        // org.springframework.validation.FieldError
        // ...
      }
    }
  }
}
```

Rendering text

```
render 'this text will be rendered back as part of the response'


render text:'<album>Revolver</album>', contentType:'text/xml'
```

Redirecting a request

```
class SampleController {

  def first() {

    // redirect to the "second" action...

    redirect action: "second"

  }

  def second() {

    // ...

  }

}
```

Redirecting a request

```
class SampleController {
  def first() {
  // redirect to the 'list' action in the 'store' controller...
  redirect action: "list", controller: "store"
  }
}
```

Redirecting a request

Argument Name	Description
action	The name of or a reference to the action to redirect to
controller	The name of the controller to redirect to
id	The id parameter to pass in the redirect
params	A map of parameters to pass
uri	A relative URI to redirect to
url	An absolute URL to redirect to

Creating a model

```
class SongController {
  def show() {
    [ song: Song.get(params.id) ]
  }
}
```

Rendering a view

```
class SongController {
  def show() {
    render view: "display", model: [ song: Song.get(params.id) ]
    //grails-app/views/song/display.gsp.
  }
}

render view:"/common/song", model:[song: Song.get(params.id) ]
//grails-app/views/common/song.gsp

render template: "/common/song", model: [song: Song.get(params.id) ]
//grails-app/views/common/_song.gsp
```

Performing data binding

```
class AlbumController {
  def save() {
    def album = new Album()
    album.genre = params.genre
    album.title = params.title
    album.save()
  }
}
```

```
class AlbumController {
  def save() {
    def album = new Album(params)
    album.save()
  }
}
```

Performing data binding

```
class AlbumController {
  def update() {
    def album = Album.get(params.id)
    album.properties = params
    album.save()
  }
}
```

The errors API and controllers

```
def save() {
  def album = Album.get(params.id)
  album.properties = params
  if(album.save()) {
    redirect action: "show", id: album.id
  } else {
    render view: "edit", model: [album:album]
  }
}
```

```
album.errors.allErrors.each { println it.code }
if(album.hasErrors()) println "Something went wrong!"
<g:renderErrors bean="${album}" />
```

Dealing with multiple objects

```
album.properties = params

<input type="text" name="title" />
```

```
<input type="text" name="album.title" />

<input type="text" name="artist.name" />


def album = new Album( params["album"] )

def artist = new Artist( params["artist"] )
```

Binding with bindData method

```
class AlbumController {
  def save() {
    def album = Album.get(params.id)
    bindData(album, params, [include:"title"])
    // ...
  }
}

bindData(album, params, [include:"title"], "album")
```

Data binding and associations

```
class Album {
  Artist artist
  // ...
}
```

```
def album = new Album(params)

<input type="text" name="artist.name" />

<input type="text" name="artist.id" value="1" />
```

```
<input type="text" name="songs[0].title" value="The Bucket" />
<input type="text" name="songs[1].title" value="Milk" />

<input type="text" name="songs[0].id" value="23" />
<input type="text" name="songs[1].id" value="47" />
```

The bindable constraint

```
class User {
  /* userName and salary would be bindable by default */
  String userName
  BigDecimal salary
  /* group and numberOfActiveGroups would not be bindable by default */
  def group
    transient int numberOfActiveGroups
    static constraints = {
      salary bindable: false
      group bindable: true
    }
}
```

Defining command objects

```
class AlbumCreateCommand {
  String artist
  String title
  List songs = []
  List durations = []
    static constraints = {
    artist blank:false
    title blank:false
    songs minSize:1, validator:{ val, obj ->
       if(val.size() != obj.durations.size())
       return "songs.durations.not.equal.size"
       }
  }
  Album createAlbum() {
    def artist = Artist.findByName(artist) ?: new Artist(name:artist)
    def album = new Album(title:title)
    songs.eachWithIndex { songTitle, i ->
      album.addToSongs(title:songTitle, duration:durations[i])
    }
  return album
  }
}
```

Using command objects

```
class AlbumController {
  def save(AlbumCreateCommand cmd) {
  // ...
  }
}

<g:form url="[controller: 'album', action: 'save'] ">
  Title: <input type="text" name="title" /> <br>
  Artist: <input type="text" name="artist" /> <br>
  Song 1: <input type="text" name="songs[0]" /> <br>
  Song 2: <input type="text" name="songs[1]" /> <br>
  ...
</g:form>
```

Using command objects

```
def save(AlbumCreateCommand cmd) {
  if(cmd.validate()) {
    def album = cmd.createAlbum()
    album.save()
    redirect(action:"show", id:album.id)
  }
  else {
    render(view:"create", model:[cmd:cmd])
  }
}
```

Imposing HTTP method restriction

```
class SongController {
  def delete() {
    if(request.method == "GET") {
      // do not delete in response to a GET request
      // redirect to the list action
      redirect action: "list"
    } else {
      // carry out the delete here...
    }
  }
}
```

Imposing HTTP method restriction

```
class SomeController {
  // action1 may be invoked via a POST
  // action2 has no restrictions
  // action3 may be invoked via a POST or DELETE
  static allowedMethods = [action1: 'POST', action3: ['POST', 'DELETE']]
  def action1() { ... }
  def action2() { ... }
  def action3() { ... }
}
```

Using simple interceptors

```
def beforeInterceptor = {
  log.trace("Executing action $actionName with params $params")
}
```

```
class AlbumController {
  private trackCountry = {
    def country = request.locale.country
    def album = Album.get(params.id)
    new AlbumVisit(country:country, album:album).save()
  }
  def beforeInterceptor = [action: trackCountry, only: 'show']
}
```

Using simple interceptors

```
class AlbumController {
  def beforeInterceptor = {
      if(request.locale != Locale.US) {
      response.sendError 403
      return false
    }
  }
}
```

```
def afterInterceptor = { model ->
  log.trace("Executed $actionName which resulted in model: $model")
}
```