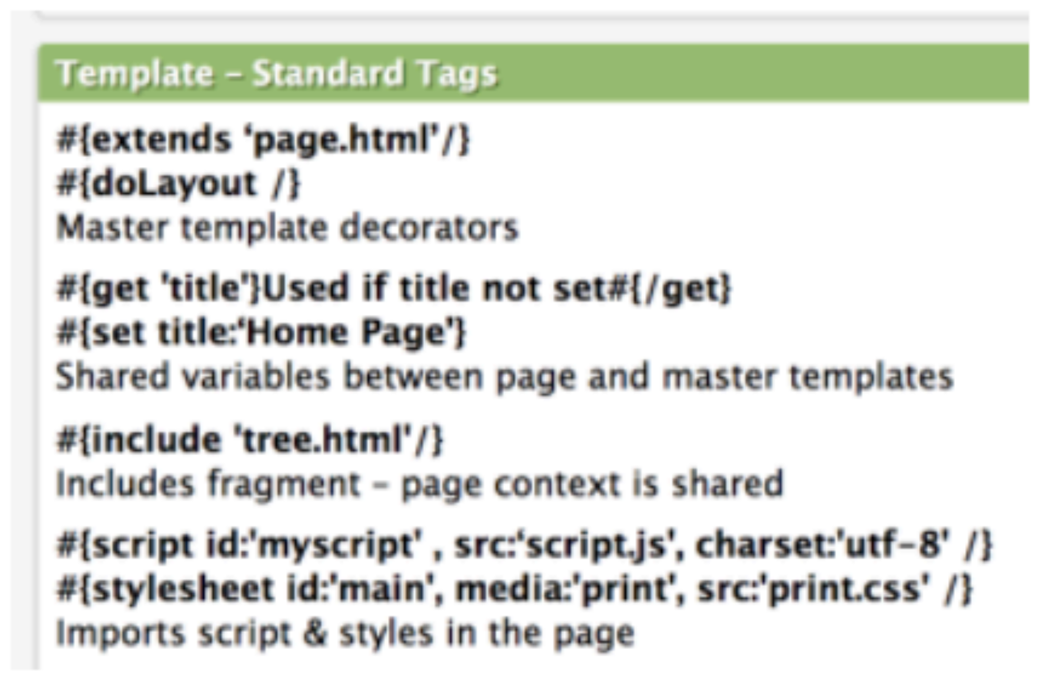
Play References

Play References + Cheat Sheet



Template - Standard Tags

#{extends 'page.html'/}
#{doLayout /}
Master template decorators

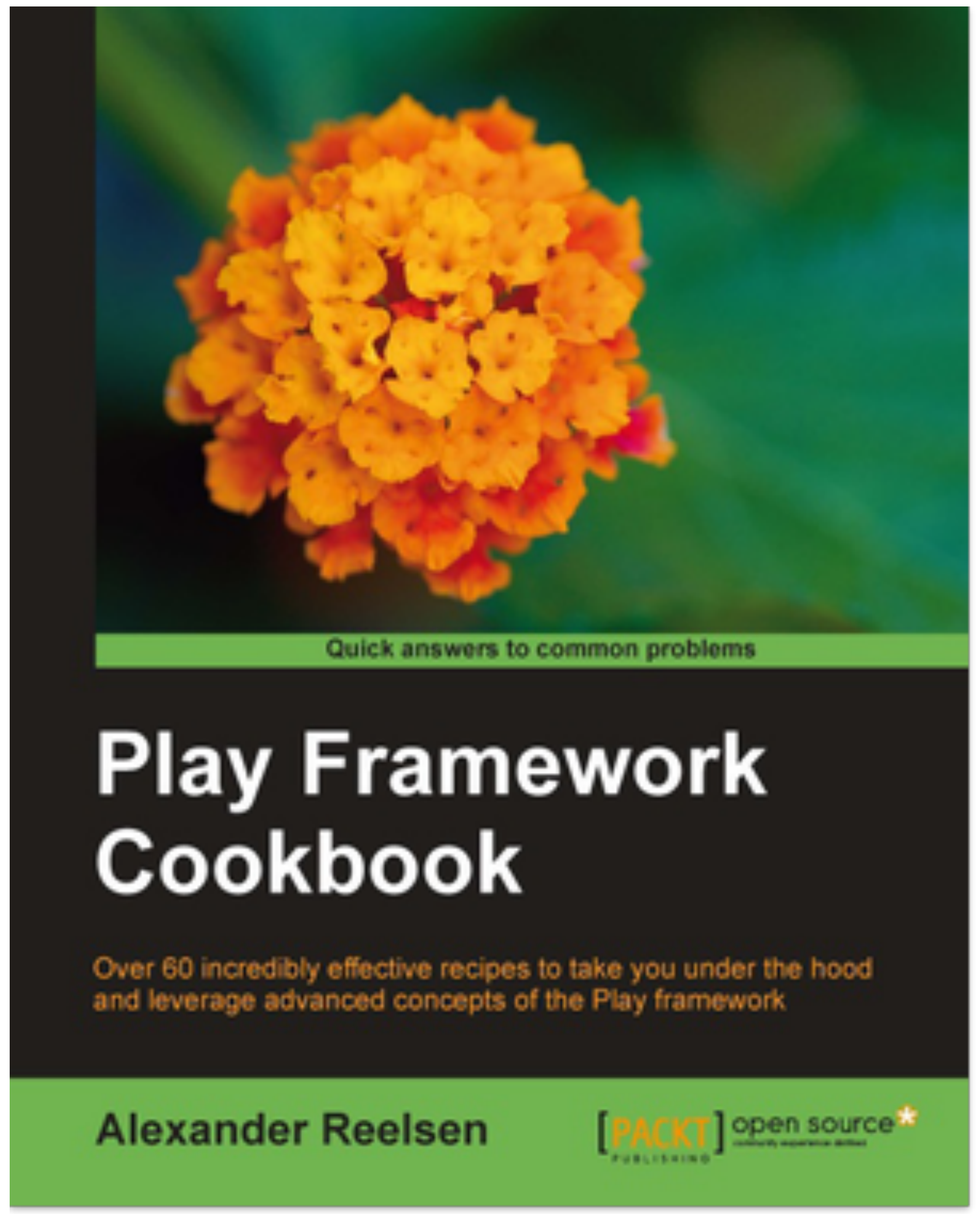
#{get 'title'}Used if title not set#{/get}
#{set title:'Home Page'}
Shared variables between page and master templates

#{include 'tree.html'/}
Includes fragment - page context is shared

#{script id:'myscript' , src:'script.js', charset:'utf-8' /}
#{stylesheet id:'main', media:'print', src:'print.css' /}
Imports script & styles in the page

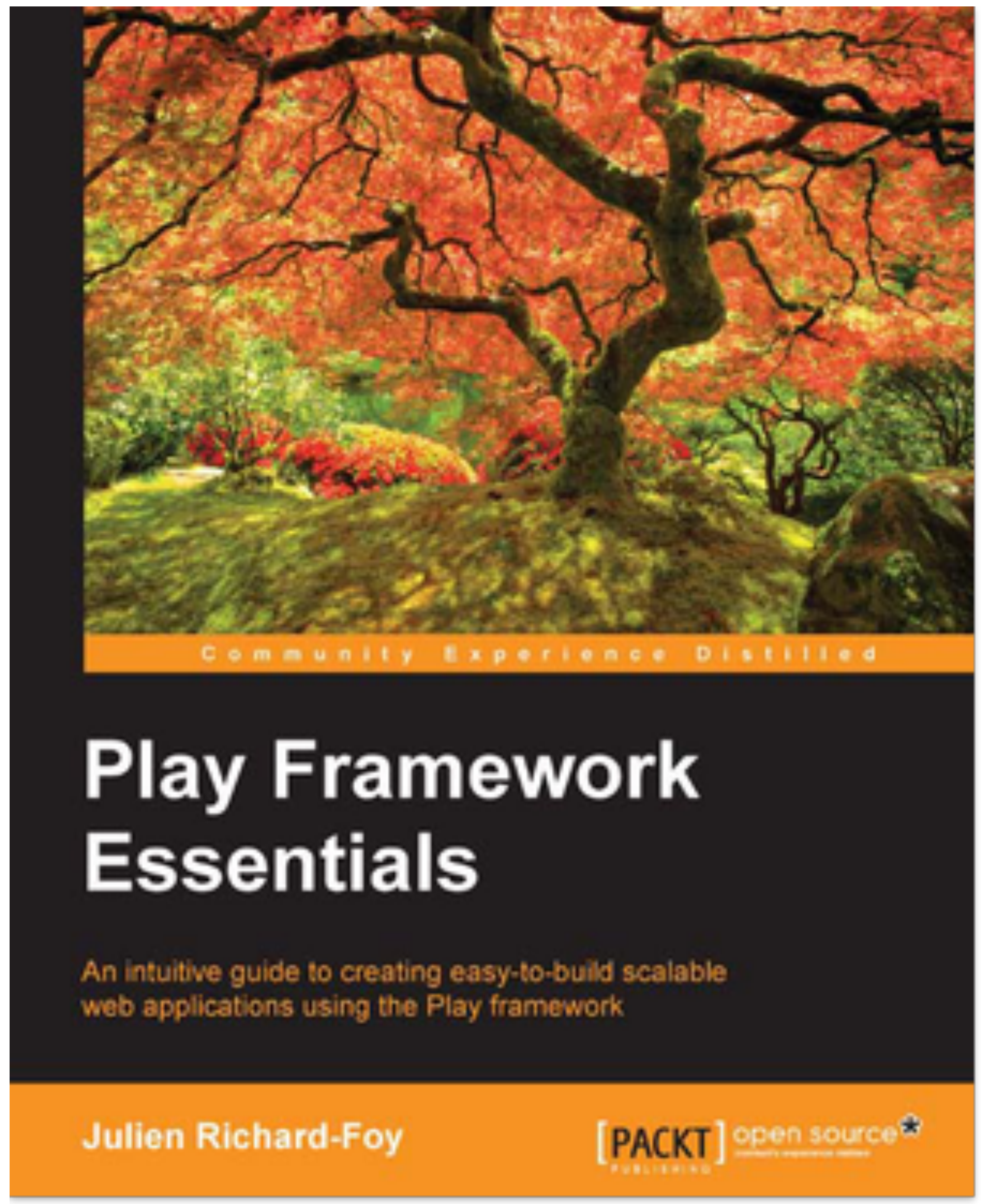
Play books & videos + Cheat Sheet

Play Framework V1.x Books


Quick answers to common problems
Play Framework Cookbook
Over 60 incredibly effective recipes to take you under the hood and leverage advanced concepts of the Play framework
Alexander Reelsen
[PACKT] open source

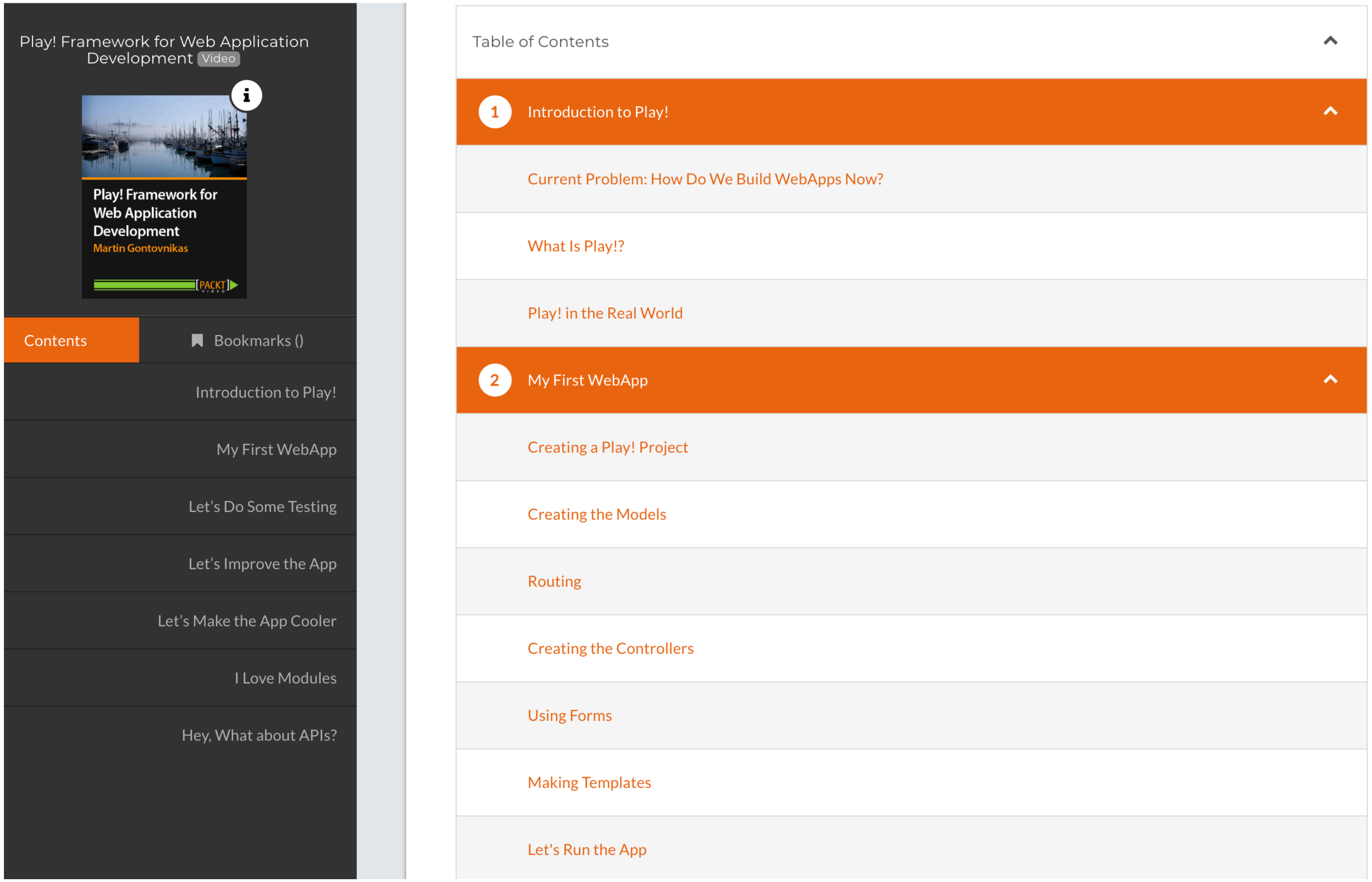
https://www.packtpub.com/web-development/play-framework-cookbook

1st Edition only, 2ed edition is for a different framework - Play 2


Community Experience Distilled
Play Framework Essentials
An intuitive guide to creating easy-to-build scalable web applications using the Play framework
Julien Richard-Foy
[PACKT] open source

https://www.packtpub.com/web-development/play-framework-essentials

Play Framework V1.x Video Course



Play! Framework for Web Application Development Video

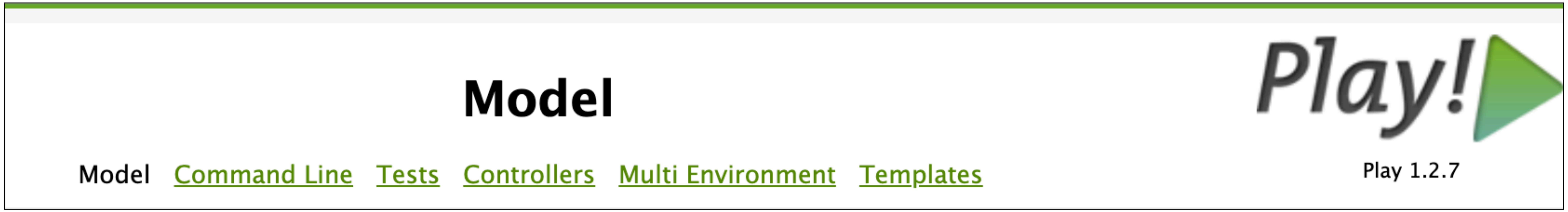
Play! Framework for Web Application Development
Martin Gontovnikas

Contents
Bookmarks ()

Table of Contents

https://www.packtpub.com/web-development/play-framework-web-application-development-video

Play Cheat Sheet

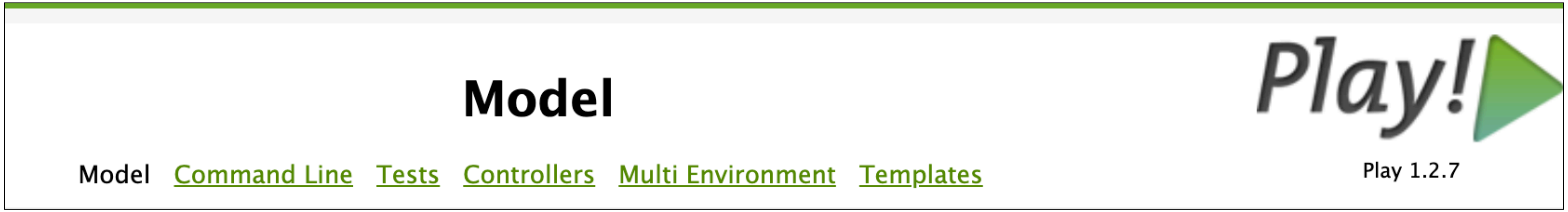


Model

Model Command Line Tests Controllers Multi Environment Templates

Play 1.2.7

https://www.playframework.com/documentation/1.2.7/cheatsheet/model



Model

Model Command Line Tests Controllers Multi Environment Templates

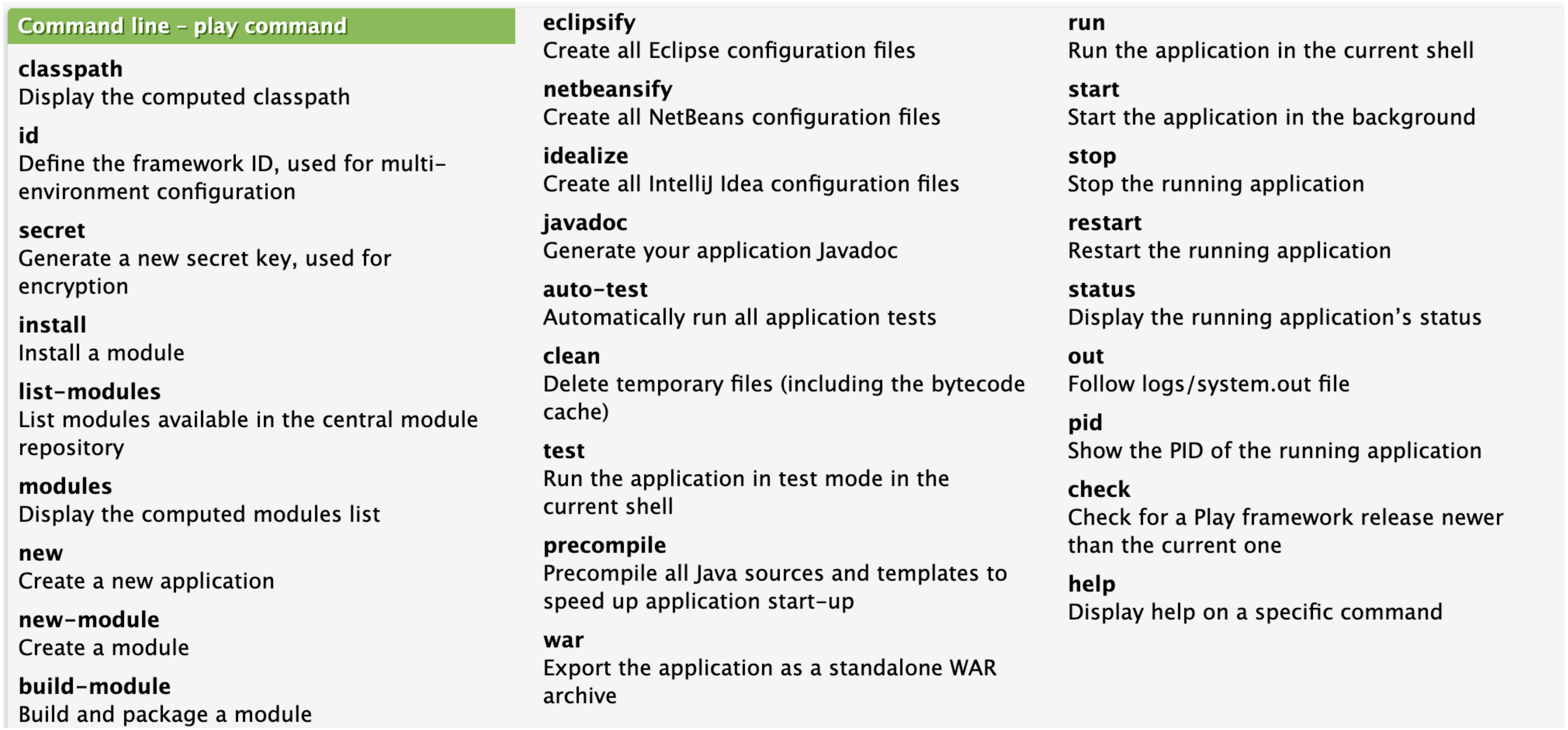
Play 1.2.7

Browse 'subset' of this subset

- Command Line
- Templates
- Models
- Controller

Command Line

Command Line



Command line – play command

classpath
Display the computed classpath

id
Define the framework ID, used for multi-environment configuration

secret
Generate a new secret key, used for encryption

install
Install a module

list-modules
List modules available in the central module repository

modules
Display the computed modules list

new
Create a new application

new-module
Create a module

build-module
Build and package a module

eclipsify
Create all Eclipse configuration files

netbeansify
Create all NetBeans configuration files

idealize
Create all IntelliJ Idea configuration files

javadoc
Generate your application Javadoc

auto-test
Automatically run all application tests

clean
Delete temporary files (including the bytecode cache)

test
Run the application in test mode in the current shell

precompile
Precompile all Java sources and templates to speed up application start-up

war
Export the application as a standalone WAR archive

run
Run the application in the current shell

start
Start the application in the background

stop
Stop the running application

restart
Restart the running application

status
Display the running application’s status

out
Follow logs/system.out file

pid
Show the PID of the running application

check
Check for a Play framework release newer than the current one

help
Display help on a specific command

Templates

Templates (1)

Template – Standard Tags

#{extends ‘page.html’/}
#{doLayout /}
Master template decorators

#{get 'title'}Used if title not set#{/get}
#{set title:‘Home Page’}
Shared variables between page and master templates

#{include 'tree.html'/}
Includes fragment – page context is shared

#{script id:'myscript' , src:‘script.js', charset:'utf-8' /}
#{stylesheet id:'main', media:'print', src:'print.css' /}
Imports script & styles in the page

#{a @Application.logout() }Disconnect#{/a}

#{form @Client.create() , id:'form' enctype:'multipart/form-data' } ... #{/form}
Handy tags to create anchors and forms

#{verbatim}${'&'}#{/verbatim}
Disables HTML escaping

#{i18n /}
Exports localized messages in Javascript

#{ifErrors} <p>Error(s) found!</p> #{/ifErrors}
Checks for validation errors

#{ifError 'user.name'} #{error 'user.name' /} #{/ifError}
Checks a given error

#{errors} <li>${error}</li> #{/errors}
Iterates over the current validation errors

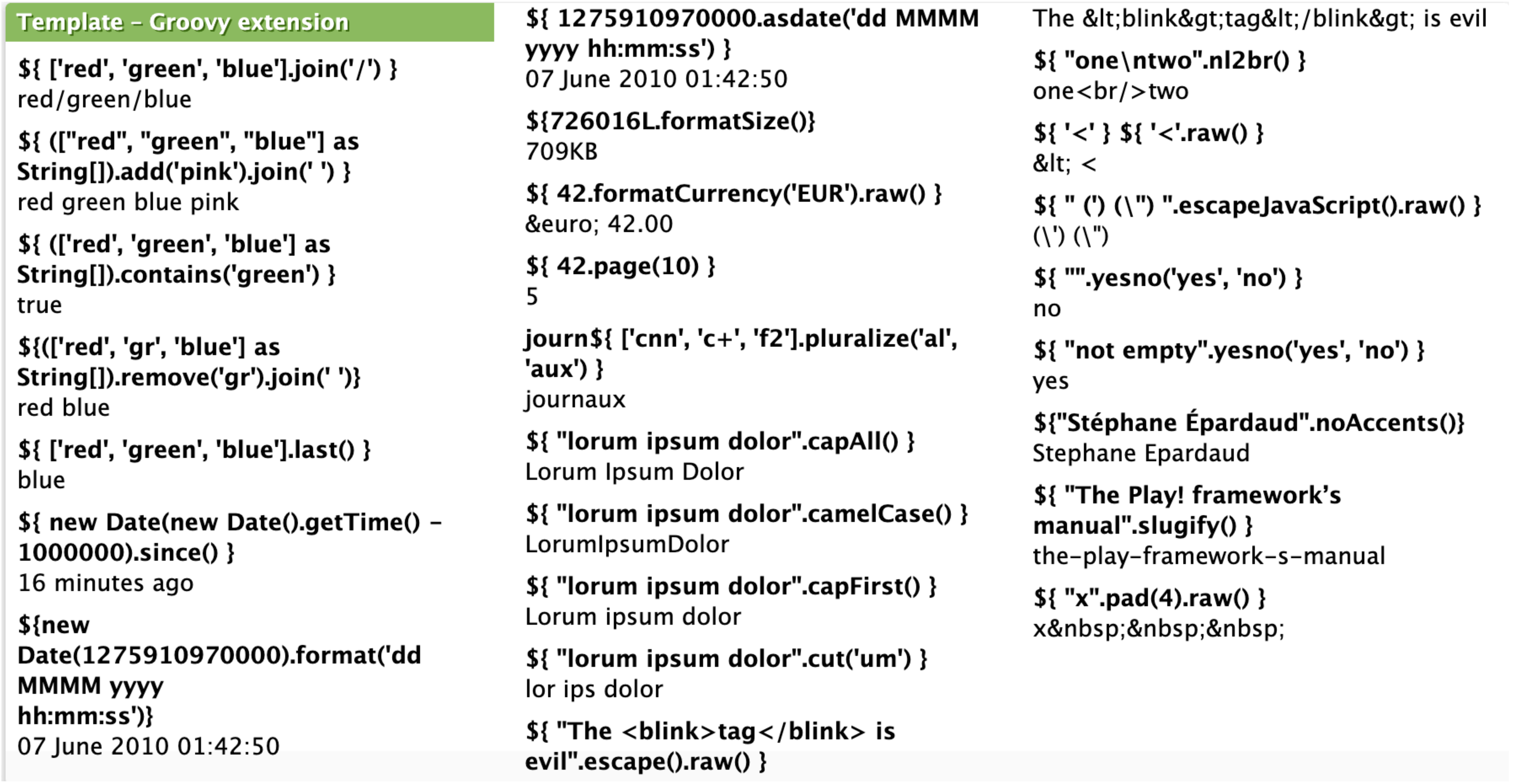
#{if cond}...#{/if}#{elseif cond}...#{/elseif}#{else}...#{/else}
#{ifnot cond}...#{/ifnot}
Conditional constructs

#{list items:0..10, as:'i'}${i}#{/list}
#{list items:'a'..'z', as:'l'}${l} ${l_isLast ?'':'|' }#{/list}
#{list users}${_}#{/list}
Loop constructs

#{list items:task, as:'task'}${task}#{/list}
#{else}No tasks on the list#{/else}
Tip: Else can be used along with list

#{cache ‘key’, for:’15min’}...#{/cache}
Caches content for 15 minutes

Templates (2)



Template – Groovy extension

${ ['red', 'green', 'blue'].join('/') }
red/green/blue

${ (["red", "green", "blue"] as String[]).add('pink').join(' ') }
red green blue pink

${ (['red', 'green', 'blue'] as String[]).contains('green') }
true

${(['red', 'gr', 'blue'] as String[]).remove('gr').join(' ')}
red blue

${ ['red', 'green', 'blue'].last() }
blue

${ new Date(new Date().getTime() - 1000000).since() }
16 minutes ago

${new Date(1275910970000).format('dd MMMM yyyy hh:mm:ss')}
07 June 2010 01:42:50

${ 1275910970000.asdate('dd MMMM yyyy hh:mm:ss') }
07 June 2010 01:42:50

${726016L.formatSize()}
709KB

${ 42.formatCurrency('EUR').raw() }
&euro; 42.00

${ 42.page(10) }
5

journ${ ['cnn', 'c+', 'f2'].pluralize('al', 'aux') }
journaux

${ "lorum ipsum dolor".capAll() }
Lorum Ipsum Dolor

${ "lorum ipsum dolor".camelCase() }
LorumIpsumDolor

${ "lorum ipsum dolor".capFirst() }
Lorum ipsum dolor

${ "lorum ipsum dolor".cut('um') }
lor ips dolor

${ "The <blink>tag</blink> is evil".escape().raw() }
The &lt;blink&gt;tag&lt;/blink&gt; is evil

${ "one\ntwo".nl2br() }
one<br/>two

${ '<' } ${ '<'.raw() }
&lt; <

${ " (') (\") ".escapeJavaScript().raw() }
(\') (\")

${ "".yesno('yes', 'no') }
no

${ "not empty".yesno('yes', 'no') }
yes

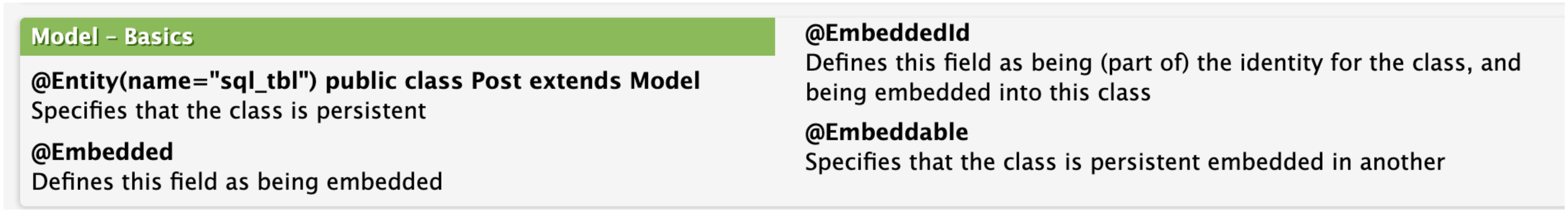
${"Stéphane Épardaud".noAccents()}
Stephane Epardaud

${ "The Play! framework’s manual".slugify() }
the-play-framework-s-manual

${ "x".pad(4).raw() }
x

Models

Models



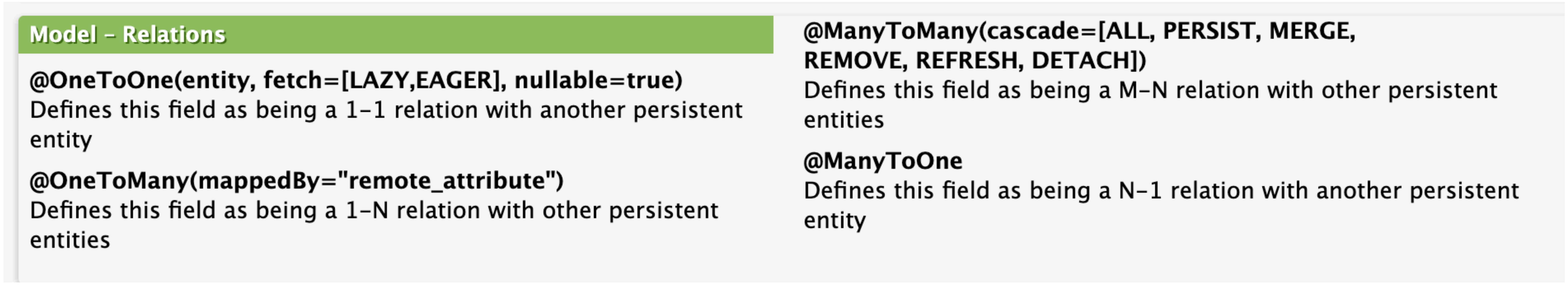
Model – Basics

@Entity(name="sql_tbl") public class Post extends Model
Specifies that the class is persistent

@Embedded
Defines this field as being embedded

@EmbeddedId
Defines this field as being (part of) the identity for the class, and being embedded into this class

@Embeddable
Specifies that the class is persistent embedded in another



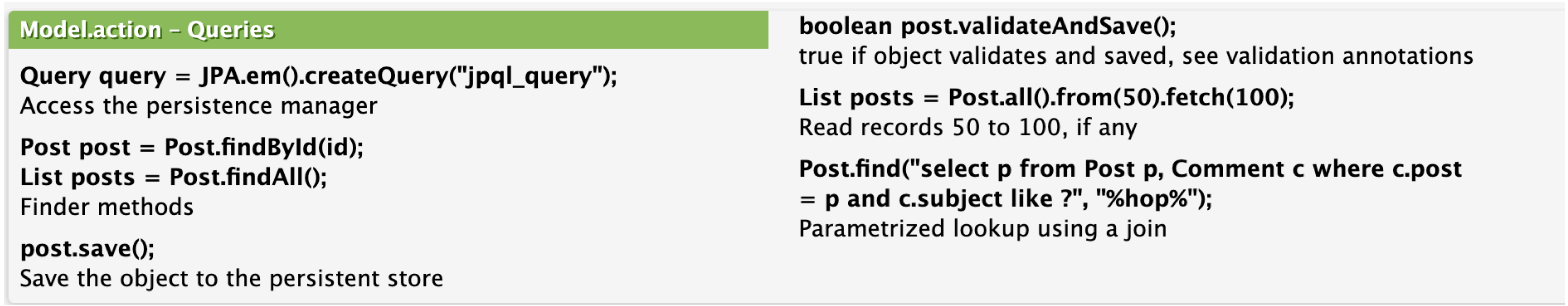
Model – Relations

@OneToOne(entity, fetch=[LAZY,EAGER], nullable=true)
Defines this field as being a 1-1 relation with another persistent entity

@OneToMany(mappedBy="remote_attribute")
Defines this field as being a 1-N relation with other persistent entities

@ManyToMany(cascade=[ALL, PERSIST, MERGE, REMOVE, REFRESH, DETACH])
Defines this field as being a M-N relation with other persistent entities

@ManyToOne
Defines this field as being a N-1 relation with another persistent entity



Model.action – Queries

Query query = JPA.em().createQuery("jpql_query");
Access the persistence manager

Post post = Post.findById(id);
List posts = Post.findAll();
Finder methods

post.save();
Save the object to the persistent store

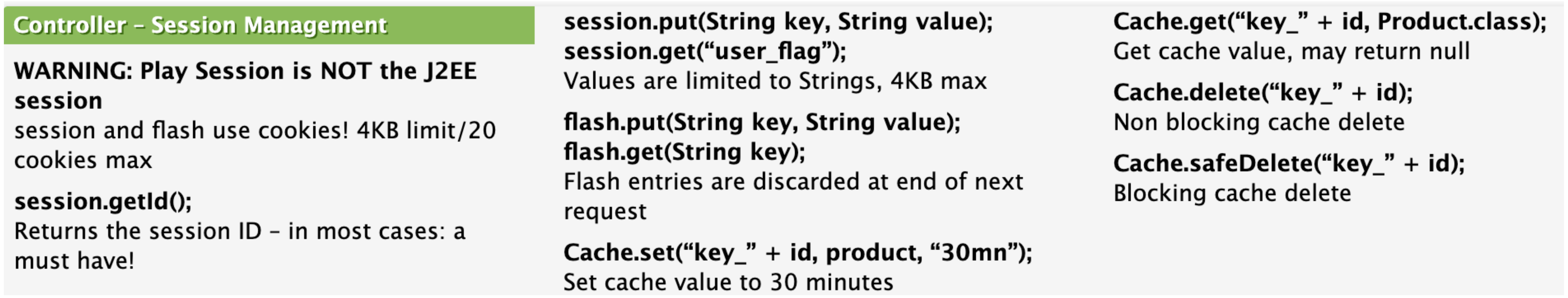
boolean post.validateAndSave();
true if object validates and saved, see validation annotations

List posts = Post.all().from(50).fetch(100);
Read records 50 to 100, if any

Post.find("select p from Post p, Comment c where c.post = p and c.subject like ?", "%hop%");
Parametrized lookup using a join

Controllers

Sessions



Controller – Session Management

WARNING: Play Session is NOT the J2EE session
session and flash use cookies! 4KB limit/20 cookies max

session.getId();
Returns the session ID – in most cases: a must have!

session.put(String key, String value);
session.get("user_flag");
Values are limited to Strings, 4KB max

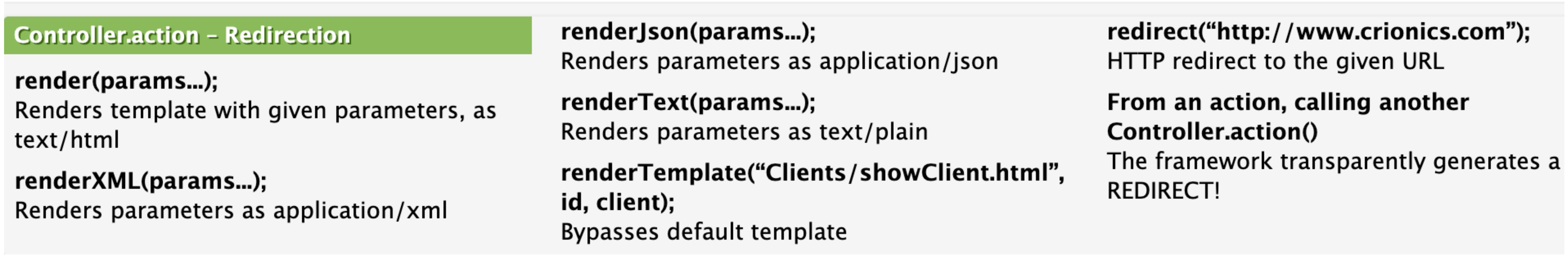
flash.put(String key, String value);
flash.get(String key);
Flash entries are discarded at end of next request

Cache.set("key_" + id, product, "30mn");
Set cache value to 30 minutes

Cache.get("key_" + id, Product.class);
Get cache value, may return null

Cache.delete("key_" + id);
Non blocking cache delete

Cache.safeDelete("key_" + id);
Blocking cache delete



Controller.action – Redirection

render(params...);
Renders template with given parameters, as text/html

renderXML(params...);
Renders parameters as application/xml

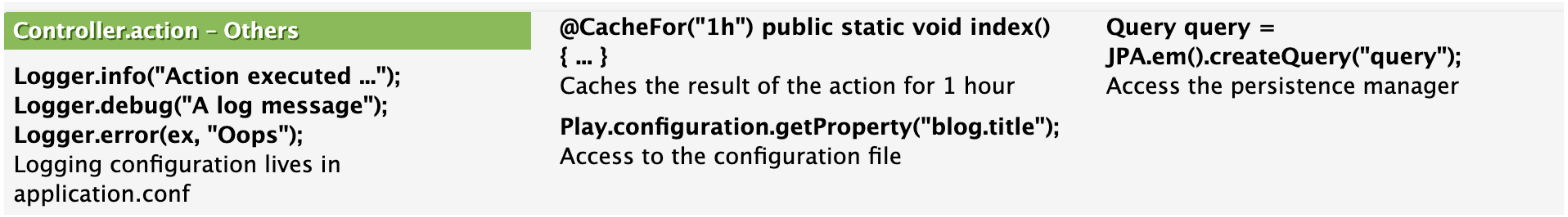
renderJson(params...);
Renders parameters as application/json

renderText(params...);
Renders parameters as text/plain

renderTemplate("Clients/showClient.html", id, client);
Bypasses default template

redirect("http://www.crionics.com");
HTTP redirect to the given URL

From an action, calling another Controller.action()
The framework transparently generates a REDIRECT!



Controller.action – Others

Logger.info("Action executed ...");
Logger.debug("A log message");
Logger.error(ex, "Oops");
Logging configuration lives in application.conf

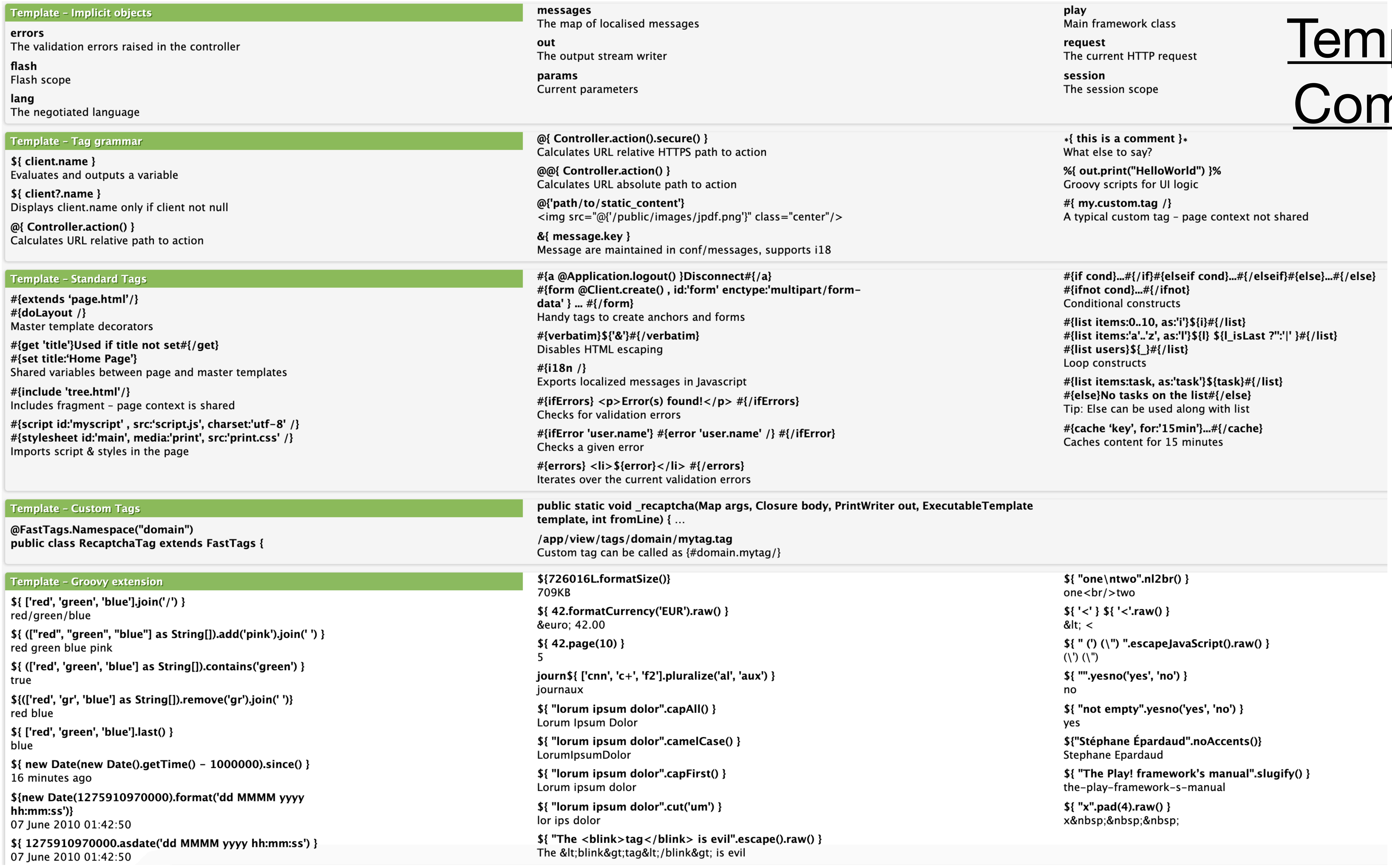
@CacheFor("1h") public static void index() { ... }
Caches the result of the action for 1 hour

Play.configuration.getProperty("blog.title");
Access to the configuration file

Query query = JPA.em().createQuery("query");
Access the persistence manager

Complete Cheat Sheets

Templates Complete



Template – Implicit objects

errors
The validation errors raised in the controller

flash
Flash scope

lang
The negotiated language

messages
The map of localised messages

out
The output stream writer

params
Current parameters

play
Main framework class

request
The current HTTP request

session
The session scope

Template – Tag grammar

${ client.name }
Evaluates and outputs a variable

${ client?.name }
Displays client.name only if client not null

@{ Controller.action() }
Calculates URL relative path to action

@{ Controller.action().secure() }
Calculates URL relative HTTPS path to action

@@{ Controller.action() }
Calculates URL absolute path to action

@{'path/to/static_content'}
<img src="@{'/public/images/jpdf.png'}" class="center"/>

&{ message.key }
Message are maintained in conf/messages, supports i18

{ this is a comment }
What else to say?

%{ out.print("HelloWorld") }%
Groovy scripts for UI logic

#{ my.custom.tag /}
A typical custom tag – page context not shared

Template – Standard Tags

#{extends ‘page.html’/}
#{doLayout /}
Master template decorators

#{get 'title'}Used if title not set#{/get}
#{set title:‘Home Page’}
Shared variables between page and master templates

#{include 'tree.html'/}
Includes fragment – page context is shared

#{script id:'myscript' , src:‘script.js', charset:'utf-8' /}
#{stylesheet id:'main', media:'print', src:'print.css' /}
Imports script & styles in the page

#{a @Application.logout() }Disconnect#{/a}
#{form @Client.create() , id:'form' enctype:'multipart/form-data' } ... #{/form}
Handy tags to create anchors and forms

#{verbatim}${'&'}#{/verbatim}
Disables HTML escaping

#{i18n /}
Exports localized messages in Javascript

#{ifErrors} <p>Error(s) found!</p> #{/ifErrors}
Checks for validation errors

#{ifError 'user.name'} #{error 'user.name' /} #{/ifError}
Checks a given error

#{errors} <li>${error}</li> #{/errors}
Iterates over the current validation errors

#{if cond}...#{/if}#{elseif cond}...#{/elseif}#{else}...#{/else}
#{ifnot cond}...#{/ifnot}
Conditional constructs

#{list items:0..10, as:'i'}${i}#{/list}
#{list items:'a'..'z', as:'l'}${l} ${l_isLast ?":'|' }#{/list}
#{list users}${_}#{/list}
Loop constructs

#{list items:task, as:'task'}${task}#{/list}
#{else}No tasks on the list#{/else}
Tip: Else can be used along with list

#{cache ‘key’, for:’15min’}...#{/cache}
Caches content for 15 minutes

Template – Custom Tags

@FastTags.Namespace("domain")
public class RecaptchaTag extends FastTags {

public static void _recaptcha(Map args, Closure body, PrintWriter out, ExecutableTemplate template, int fromLine) { ...

/app/view/tags/domain/mytag.tag
Custom tag can be called as {#domain.mytag/}

Template – Groovy extension

${ ['red', 'green', 'blue'].join('/') }
red/green/blue

${ (["red", "green", "blue"] as String[]).add('pink').join(' ') }
red green blue pink

${ (['red', 'green', 'blue'] as String[]).contains('green') }
true

${(['red', 'gr', 'blue'] as String[]).remove('gr').join(' ')}
red blue

${ ['red', 'green', 'blue'].last() }
blue

${ new Date(new Date().getTime() - 1000000).since() }
16 minutes ago

${new Date(1275910970000).format('dd MMMM yyyy hh:mm:ss')}
07 June 2010 01:42:50

${ 1275910970000.asdate('dd MMMM yyyy hh:mm:ss') }
07 June 2010 01:42:50

${726016L.formatSize()}
709KB

${ 42.formatCurrency('EUR').raw() }
&euro; 42.00

${ 42.page(10) }
5

journ${ ['cnn', 'c+', 'f2'].pluralize('al', 'aux') }
journaux

${ "lorum ipsum dolor".capAll() }
Lorum Ipsum Dolor

${ "lorum ipsum dolor".camelCase() }
LorumIpsumDolor

${ "lorum ipsum dolor".capFirst() }
Lorum ipsum dolor

${ "lorum ipsum dolor".cut('um') }
lor ips dolor

${ "The <blink>tag</blink> is evil".escape().raw() }
The &lt;blink&gt;tag&lt;/blink&gt; is evil

${ "one\ntwo".nl2br() }
one<br/>two

${ '<' } ${ '<'.raw() }
&lt; <

${ " (') (\") ".escapeJavaScript().raw() }
(\') (\")

${ "".yesno('yes', 'no') }
no

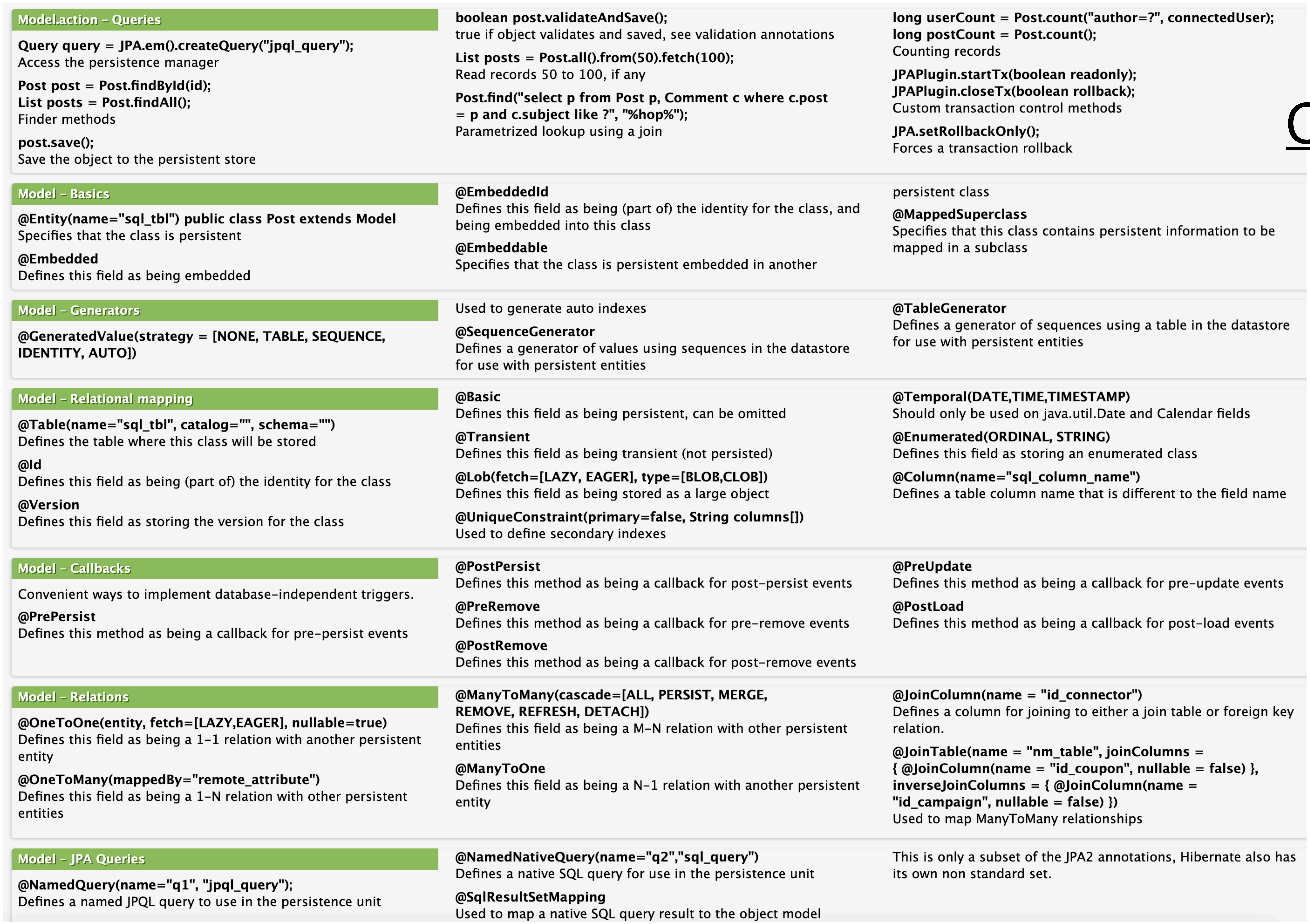
${ "not empty".yesno('yes', 'no') }
yes

${"Stéphane Épardaud".noAccents()}
Stephane Epardaud

${ "The Play! framework’s manual".slugify() }
the-play-framework-s-manual

${ "x".pad(4).raw() }
x

Models Complete



Model.action – Queries

Query query = JPA.em().createQuery("jpql_query");
Access the persistence manager

Post post = Post.findById(id);
List posts = Post.findAll();
Finder methods

post.save();
Save the object to the persistent store

boolean post.validateAndSave();
true if object validates and saved, see validation annotations

List posts = Post.all().from(50).fetch(100);
Read records 50 to 100, if any

Post.find("select p from Post p, Comment c where c.post = p and c.subject like ?", "%hop%");
Parametrized lookup using a join

long userCount = Post.count("author=?", connectedUser);
long postCount = Post.count();
Counting records

JPAPlugin.startTx(boolean readonly);
JPAPlugin.closeTx(boolean rollback);
Custom transaction control methods

JPA.setRollbackOnly();
Forces a transaction rollback

Model – Basics

@Entity(name="sql_tbl") public class Post extends Model
Specifies that the class is persistent

@Embedded
Defines this field as being embedded

@EmbeddedId
Defines this field as being (part of) the identity for the class, and being embedded into this class

@Embeddable
Specifies that the class is persistent embedded in another persistent class

@MappedSuperclass
Specifies that this class contains persistent information to be mapped in a subclass

Model – Generators

@GeneratedValue(strategy = [NONE, TABLE, SEQUENCE, IDENTITY, AUTO])
Used to generate auto indexes

@SequenceGenerator
Defines a generator of values using sequences in the datastore for use with persistent entities

@TableGenerator
Defines a generator of sequences using a table in the datastore for use with persistent entities

Model – Relational mapping

@Table(name="sql_tbl", catalog="", schema="")
Defines the table where this class will be stored

@Id
Defines this field as being (part of) the identity for the class

@Version
Defines this field as storing the version for the class

@Basic
Defines this field as being persistent, can be omitted

@Transient
Defines this field as being transient (not persisted)

@Lob(fetch=[LAZY, EAGER], type=[BLOB,CLOB])
Defines this field as being stored as a large object

@UniqueConstraint(primary=false, String columns[])
Used to define secondary indexes

@Temporal(DATE,TIME,TIMESTAMP)
Should only be used on java.util.Date and Calendar fields

@Enumerated(ORDINAL, STRING)
Defines this field as storing an enumerated class

@Column(name="sql_column_name")
Defines a table column name that is different to the field name

Model – Callbacks

Convenient ways to implement database-independent triggers.

@PrePersist
Defines this method as being a callback for pre-persist events

@PostPersist
Defines this method as being a callback for post-persist events

@PreRemove
Defines this method as being a callback for pre-remove events

@PostRemove
Defines this method as being a callback for post-remove events

@PreUpdate
Defines this method as being a callback for pre-update events

@PostLoad
Defines this method as being a callback for post-load events

Model – Relations

@OneToOne(entity, fetch=[LAZY,EAGER], nullable=true)
Defines this field as being a 1-1 relation with another persistent entity

@OneToMany(mappedBy="remote_attribute")
Defines this field as being a 1-N relation with other persistent entities

@ManyToMany(cascade=[ALL, PERSIST, MERGE, REMOVE, REFRESH, DETACH])
Defines this field as being a M-N relation with other persistent entities

@ManyToOne
Defines this field as being a N-1 relation with another persistent entity

@JoinColumn(name = "id_connector")
Defines a column for joining to either a join table or foreign key relation.

@JoinTable(name = "nm_table", joinColumns = { @JoinColumn(name = "id_coupon", nullable = false) }, inverseJoinColumns = { @JoinColumn(name = "id_campaign", nullable = false) })
Used to map ManyToMany relationships

Model – JPA Queries

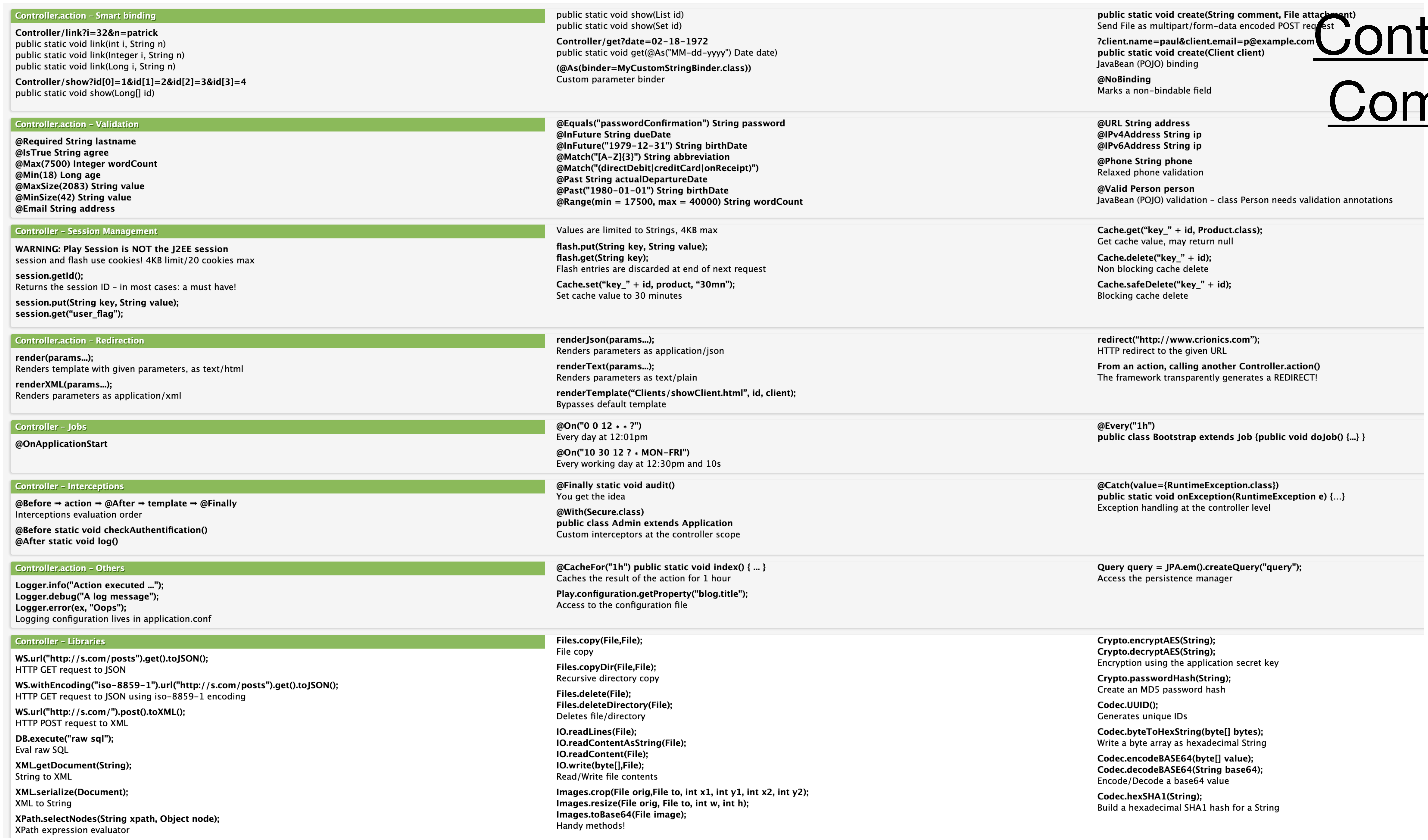
@NamedQuery(name="q1", "jpql_query");
Defines a named JPQL query to use in the persistence unit

@NamedNativeQuery(name="q2","sql_query")
Defines a native SQL query for use in the persistence unit

@SqlResultSetMapping
Used to map a native SQL query result to the object model

This is only a subset of the JPA2 annotations, Hibernate also has its own non standard set.

Controllers Complete



Controller.action - Smart binding

Controller/link?i=32&n=patrick
public static void link(int i, String n)
public static void link(Integer i, String n)
public static void link(Long i, String n)

Controller/show?id[0]=1&id[1]=2&id[2]=3&id[3]=4
public static void show(Long[] id)

public static void show(List id)
public static void show(Set id)

Controller/get?date=02-18-1972
public static void get(@As("MM-dd-yyyy") Date date)

(@As(binder=MyCustomStringBinder.class))
Custom parameter binder

public static void create(String comment, File attachment)
Send File as multipart/form-data encoded POST request

?client.name=paul&client.email=p@example.com
public static void create(Client client)
JavaBean (POJO) binding

@NoBinding
Marks a non-bindable field

Controller.action - Validation

@Required String lastname
@IsTrue String agree
@Max(7500) Integer wordCount
@Min(18) Long age
@MaxSize(2083) String value
@MinSize(42) String value
@Email String address

@Equals("passwordConfirmation") String password
@InFuture String dueDate
@InFuture("1979-12-31") String birthDate
@Match("[A-Z]{3}") String abbreviation
@Match("(directDebit|creditCard|onReceipt)")
@Past String actualDepartureDate
@Past("1980-01-01") String birthDate
@Range(min = 17500, max = 40000) String wordCount

@URL String address
@IPv4Address String ip
@IPv6Address String ip

@Phone String phone
Relaxed phone validation

@Valid Person person
JavaBean (POJO) validation - class Person needs validation annotations

Controller - Session Management

WARNING: Play Session is NOT the J2EE session
session and flash use cookies! 4KB limit/20 cookies max

session.getId();
Returns the session ID - in most cases: a must have!

session.put(String key, String value);
session.get("user_flag");

Values are limited to Strings, 4KB max

flash.put(String key, String value);
flash.get(String key);
Flash entries are discarded at end of next request

Cache.set("key_" + id, product, "30mn");
Set cache value to 30 minutes

Cache.get("key_" + id, Product.class);
Get cache value, may return null

Cache.delete("key_" + id);
Non blocking cache delete

Cache.safeDelete("key_" + id);
Blocking cache delete

Controller.action - Redirection

render(params...);
Renders template with given parameters, as text/html

renderXML(params...);
Renders parameters as application/xml

renderJson(params...);
Renders parameters as application/json

renderText(params...);
Renders parameters as text/plain

renderTemplate("Clients/showClient.html", id, client);
Bypasses default template

redirect("http://www.crionics.com");
HTTP redirect to the given URL

From an action, calling another Controller.action()
The framework transparently generates a REDIRECT!

Controller - Jobs

@OnApplicationStart

@On("0 0 12 * * ?")
Every day at 12:01pm

@On("10 30 12 ? * MON-FRI")
Every working day at 12:30pm and 10s

@Every("1h")
public class Bootstrap extends Job {public void doJob() {...} }

Controller - Interceptions

@Before ➡ action ➡ @After ➡ template ➡ @Finally
Interceptions evaluation order

@Before static void checkAuthentification()
@After static void log()

@Finally static void audit()
You get the idea

@With(Secure.class)
public class Admin extends Application
Custom interceptors at the controller scope

@Catch(value={RuntimeException.class})
public static void onException(RuntimeException e) {...}
Exception handling at the controller level

Controller.action - Others

Logger.info("Action executed ...");
Logger.debug("A log message");
Logger.error(ex, "Oops");
Logging configuration lives in application.conf

@CacheFor("1h") public static void index() { ... }
Caches the result of the action for 1 hour

Play.configuration.getProperty("blog.title");
Access to the configuration file

Query query = JPA.em().createQuery("query");
Access the persistence manager

Controller - Libraries

WS.url("http://s.com/posts").get().toJSON();
HTTP GET request to JSON

WS.withEncoding("iso-8859-1").url("http://s.com/posts").get().toJSON();
HTTP GET request to JSON using iso-8859-1 encoding

WS.url("http://s.com/").post().toXML();
HTTP POST request to XML

DB.execute("raw sql");
Eval raw SQL

XML.getDocument(String);
String to XML

XML.serialize(Document);
XML to String

XPath.selectNodes(String xpath, Object node);
XPath expression evaluator

Files.copy(File,File);
File copy

Files.copyDir(File,File);
Recursive directory copy

Files.delete(File);
Files.deleteDirectory(File);
Deletes file/directory

IO.readLines(File);
IO.readContentAsString(File);
IO.readContent(File);
IO.write(byte[],File);
Read/Write file contents

Images.crop(File orig,File to, int x1, int y1, int x2, int y2);
Images.resize(File orig, File to, int w, int h);
Images.toBase64(File image);
Handy methods!

Crypto.encryptAES(String);
Crypto.decryptAES(String);
Encryption using the application secret key

Crypto.passwordHash(String);
Create an MD5 password hash

Codec.UUID();
Generates unique IDs

Codec.byteToHexString(byte[] bytes);
Write a byte array as hexadecimal String

Codec.encodeBASE64(byte[] value);
Codec.decodeBASE64(String base64);
Encode/Decode a base64 value

Codec.hexSHA1(String);
Build a hexadecimal SHA1 hash for a String