2022-02-28

Catala Syntax Cheat Sheet

v.0.5.0

Literate programming

Feature	English syntax	French syntax
Heading	`## Foo` `### Bar`	`## Foo` `### Bar`
Code block	```` ```catala ```` ```` ``` ````	```` ```catala ```` ```` ``` ````
Metadata block	```` ```catala-metadata ```` ```` ``` ````	```` ```catala-metadata ```` ```` ``` ````
File inclusion	`> Include: foo.catala_en`	`> Inclusion: foo.catala_fr`

Literals

Feature	English syntax	French syntax
Integers	`65536`	`65536`
Decimals	`65536.262144`	`65536.262144`
Money	`$1,234,567.89`	`1 234 567,89 €`
Date	`|2021-01-31|`	`|2021-31-01|`
Durations	`254 day 4 month 1 year`	`254 jour 4 mois 1 an`
Boolean	`true` `false`	`vrai` `faux`

Metadata declaration

Feature	English syntax	French syntax

Structure declaration	`declaration structure Foo:`
	`  data bar content integer`
	`  data baz content boolean`
		`déclaration structure Foo:`
		`  donnée bar contenu entier`
		`  donnée baz contenu booléen`

Enumeration declaration	`declaration enumeration Foo:`
	`  -- Bar content integer`
	`  -- Baz`
		`déclaration énumeration Foo:`
		`  -- Bar contenu entier`
		`  -- Baz`

Scope declaration	`declaration scope Foo:`
	`  internal bar content integer`
	`  internal baz condition`
	`  fizz scope Buzz`
		`déclaration champ d'application Foo:`
		`  interne bar contenu entier`
		`  interne baz condition`
		`  contexte fizz champ d'application Buzz`

Input-output qualifiers	`internal bar content ...`
	`output baz content ...`
	`input fizz content ...`
	`input output buzz content ...`
	`context biz content ...`
	`context output boz content ...`
		`interne bar contenu ...`
		`sortie baz contenu ...`
		`entrée fizz contenu ...`
		`entrée sortie buzz contenu ...`
		`contexte biz contenu ...`
		`contexte sortie boz contenu ...`

State transitions declaration	`internal foo content ...`
	`  state bar`
	`  state buzz`
		`interne foo contenu ...`
		`  état bar`
		`  état buzz`

Scope use and related items

Feature	English syntax	French syntax
Scope use	`scope Foo: ...`	`champ d'application Foo: ...`
Use-wide condition	`scope Foo` `  under condition bar: ...`	`champ d'application Foo` `  sous condition bar: ...`
Unconditional definition	`definition foo equals ...`	`définition foo égal à ...`
Conditional definition	`definition foo under condition` `  bar consequence equals ...`	`définition foo sous condition` `  bar conséquence égal à ...`
Rule (definition for conditions)	`rule foo under condition` `  bar consequence fulfilled`	`règle foo sous condition` `  bar conséquence rempli`
Negative rule	`rule foo under condition` `  bar consequence not fulfilled`	`règle foo sous condition` `  bar conséquence non rempli`
Function definition/rule	`definition foo of bar ...`	`définition foo de bar ...`
Labeled definition or rule	`label foo definition bar ...`	`étiquette foo définition bar ...`
Exception to label	`exception foo definition bar ...`	`exception foo définition bar ...`
Exception to implicit	`exception definition bar ...`	`exception définition bar ...`
State definition	`definition foo state bar equals ...`	`définition foo état bar égal à ...`
Assertion	`assertion...`	`assertion ...`

Types

Feature	English syntax	French syntax
Natural integers	`integer`	`entier`
Rational numbers	`decimal`	`décimal`
Booleans	`boolean`	`booléen`
Money	`money`	`argent`
Date	`date`	`date`
Duration	`duration`	`durée`
Function	`Foo depends on Bar`	`Foo dépend de Bar`
Collection	`collection Foo`	`collection Foo`

Expressions

Feature	English syntax	French syntax

Pattern matching	`match ... with pattern`
	`  -- Foo of foo: ...`
	`  -- Bar : ...`
		`selon ... sous forme`
		`  -- Foo de foo: ...`
		`  -- Bar: ...`

Pattern test and optional binding	`... with pattern Foo` `... with pattern Bar of bar and ...`	`... sous forme Foo` `... sous forme Bar de bar et`
Constructor injection	`Foo content ... Bar`	`Foo contenu ... Bar`
Structure literal	`Foo { -- bar: ... -- baz: ...}`	`Foo { -- bar: ... -- baz: ...}`
Structure field access	`(...).foo`	`(...).foo`
Function call	`... of ...`	`... de ...`
Subscope variable	`foo.bar`	`foo.bar`
Conditional	`if ... then ... else ...`	`si ... alors ... sinon`

Collections

Feature	English syntax	French syntax
Collection literal	`[ ...; ...; ...]`	`[ ...; ...; ...]`
Presence test	`... in ...`	`... dans ...`
Cardinal	`number of ...`	`nombre de ...`
Existence test	`exists foo in ... such that ...`	`existe foo dans ... tel que ...`
For all test	`for all foo in ... we have ...`	`pour tout foo dans ... on a ...`
For all test	`for all foo in ... we have ...`	`pour tout foo dans ... on a ...`
Map/filter	`map for foo in ... of ...` `filter for foo in ... of ...`	`application pour foo dans ... de ...` `filtre pour foo dans ... de ...`
Aggregation	`sum money for foo in ... of ...`	`somme argent pour foo dans ... de ...`
Conditional count	`number for foo in ... of ...`	`nombre pour foo dans ... de ...`
Extremum	`maximum integer initial ... for ... in ...`	`maximum entier initial ... pour ... dans ...`
Arg-extremum	`content minimum decimal initial ... for ... in ...`	`contenu minimum décimal initial ... pour ... dans ...`

Operators

Feature	English syntax	French syntax
Integer to decimal	`integer_to_decimal of ...`	`entier_vers_décimal de ...`
Money rounding	`round_money of ...`	`arrondi_argent de ...`

Date parts	`get_day of ...`
	`get_month of ...`
	`get_year of ...`
		`accès_jour de ...`
		`accès_mois de ...`
		`accès_année de ...`

Logical inclusive or	`... or ...`	`... ou ...`
Logical exclusive or	`... xor ...`	`... ou bien ...`
Logical and	`... and ...`	`... et ...`
Polymorphic structural equality	`... = ...` `... != ...`	`... = ...` `... != ...`
Integer sum	`(integer) + (integer)`	`(entier) + (entier)`
Integer substraction	`(integer) - (integer)`	`(entier) - (entier)`
Integer multiplication	`(integer) * (integer)`	`(entier) * (entier)`
Integer division	`(integer) / (integer)`	`(entier) / (entier)`
Integer comparison	`< <= > >=`	`< <= > >=`
Decimal sum	`(decimal) +. (decimal)`	`(décimal) +. (décimal)`
Decimal substraction	`(decimal) -. (decimal)`	`(décimal) -. (décimal)`
Decimal multiplication	`(decimal) *. (decimal)`	`(décimal) *. (décimal)`
Decimal division	`(decimal) /. (decimal)`	`(décimal) /. (décimal)`
Decimal comparison	`<. <=. >. >=.`	`<. <=. >. >=.`
Money sum	`(money) +$ (money)`	`(argent) +€ (argent)`
Money substraction	`(money) -$ (money)`	`(argent) -€ (argent)`
Money multiplication	`(money) *$ (decimal)`	`(argent) *€ (décimal)`
Money division	`(money) /$ (money)`	`(argent) /€ (argent)`
Money comparison	`<$ <=$ >$ >=$`	`<€ <=€ >€ >=€`
Date sum	`(date) +@ (duration)`	`(date) +@ (durée)`
Date substraction	`(date) -@ (date)`	`(date) -@ (date)`
Date comparison	`<@ <=@ >@ >=@`	`<@ <=@ >@ >=@`
Duration sum	`(duration) +^ (duration)`	`(durée) +^ (durée)`
Duration substraction	`(duration) -^ (duration)`	`(durée) -^ (durée)`
Duration division	`(duration) /^ (duration)`	`(durée) /^ (durée)`
Duration comparison	`<^ <=^ >^ >=^`	`<^ <=^ >^ >=^`