Lekta framework practical tutorial

Jose F Quesada & Jose Luis Pro

What is Lekta?

Lekta is a software **framework** oriented to the design and implementation of Natural Language Processing (NLP) related applications. This includes:

- Some specialized and **optimized** modules widely used in NLP applications (tokenizer, parser and so on). You'll never reinvent the wheel anymore.
- A simple and efficient way to define lexicons and grammar rules for any natural language.
- Early multilingual support for all your applications.
- A set of built-in functions that you'll find useful when implementing your NLP oriented app.
- A programming language to interact with all items above and to define your own functions or procedures.

What is Lekta?

Lekta is a software **framework** oriented to the design and implementation of Natural Language Processing (NLP) related applications. This includes:

- Some specialized and **optimized** modules widely used in NLP applications (tokenizer, parser and so on). You'll never reinvent the wheel anymore.
- A simple and efficient way to define lexicons and grammar rules for any natural language.
- Early multilingual support for all your applications.
- A set of built-in functions that you'll find useful when implementing your NLP oriented app.
- A programming language to interact with all items above and to define your own functions or procedures.

What is Lekta?

Lekta is a software **framework** oriented to the design and implementation of Natural Language Processing (NLP) related applications. This includes:

- Some specialized and **optimized** modules widely used in NLP applications (tokenizer, parser and so on). You'll never reinvent the wheel anymore.
- A simple and efficient way to define lexicons and grammar rules for any natural language.
- Early multilingual support for all your applications.
- A set of built-in functions that you'll find useful when implementing your NLP oriented app.
- A programming language to interact with all items above and to define your own functions or procedures.

What is Lekta?

Lekta is a software **framework** oriented to the design and implementation of Natural Language Processing (NLP) related applications. This includes:

- Some specialized and **optimized** modules widely used in NLP applications (tokenizer, parser and so on). You'll never reinvent the wheel anymore.
- A simple and efficient way to define lexicons and grammar rules for any natural language.
- Early multilingual support for all your applications.
- A set of built-in functions that you'll find useful when implementing your NLP oriented app.
- A programming language to interact with all items above and to define your own functions or procedures.

What is Lekta?

Lekta is a software **framework** oriented to the design and implementation of Natural Language Processing (NLP) related applications. This includes:

- Some specialized and **optimized** modules widely used in NLP applications (tokenizer, parser and so on). You'll never reinvent the wheel anymore.
- A simple and efficient way to define lexicons and grammar rules for any natural language.
- Early multilingual support for all your applications.
- A set of built-in functions that you'll find useful when implementing your NLP oriented app.
- A programming language to interact with all items above and to define your own functions or procedures.

What is Lekta?

Lekta is a software **framework** oriented to the design and implementation of Natural Language Processing (NLP) related applications. This includes:

- Some specialized and **optimized** modules widely used in NLP applications (tokenizer, parser and so on). You'll never reinvent the wheel anymore.
- A simple and efficient way to define lexicons and grammar rules for any natural language.
- Early multilingual support for all your applications.
- A set of built-in functions that you'll find useful when implementing your NLP oriented app.
- A programming language to interact with all items above and to define your own functions or procedures.

One file .lkt with at least 5 sections and one file .slk

AnBm.lkt

```
// ***************************************************************
//
// Exercise 01: Generator/Recognizer for language AnBm. Where n,m >= 1
//
// ***************************************************************

lektaProject
   projectHead
      projectLanguageScope : [ anbm ]
      projectCompileOutput : ".AnBm.olk"

   projectSetup
      setupParserRoots = S

   classModel
      classDef:Void ( S, A, B, a, b )

   lexicalModel forLanguage anbm
      ("a", a)
      ("b", b)

   grammaticalModel forLanguage anbm
      (R1: [ S -> a A b B ])
      (R2: [ A -> ])
      (R3: [ A -> a A ])
      (R4: [ B -> ])
      (R5: [ B -> b B ])
```

Programming structures: comments

```
// This is a mono-line comment
```

```
/* This is a multi-line comment
      with some lines
      commented */
```

Programming structures: if...else if...else

```
if(cond1)
{
    // Body 1
}
else if(cond2)
{
    // Body 2
}
...
else
{
    // Body n
}
```

Programming structures: switch

```
string GetMonthName(integer month) {
    switch (month) {
        case  1 { return 'January';}
        case  2 { return 'February';}
        case  3 { return 'March';}
        case  4 { return 'April';}
        case  5 { return 'May';}
        case  6 { return 'June';}
        case  7 { return 'July';}
        case  8 { return 'August';}
        case  9 { return 'September';}
        case 10 { return 'October';}
        case 11 { return 'November';}
        case 12 { return 'December';}
    }
```

Programming structures: cond

```
cond{
    (!!cal.CalendarDay) {
        errorMessage <- 'The month day of
           the provided date is missing.';
    }
    (!!cal.CalendarMonth) {
        errorMessage <- 'The month of the
           provided date is missing.';
    }
    (!!cal.CalendarYear) {
        errorMessage <- 'The year of the
           provided date is missing.';
    }
      /***** TODO
    default {
```

Programming structures: while

```
// TODO
```

Programming structures: for

```
// TODO
```

Opperators:

```
// TODO
```

Meta-types