Artigo do Trabalho de Graduação

Cristhiane Resende; Suellen Portugal & Tiso Puccinelli
Orientador(es): Prof. Me. Nelson Augusto Oliveira de Aguiar

Graduandos do Curso de Ciência da Computação da USJT, turma de 2015.

Título: Artigo do Trabalho de Graduação: – Tradutor de Regex
Resumo: Neste artigo será apresentada a ferramenta Tradutor de Regex, criada como uma aplicação web para facilitar a interpretação e o entendimento de expressões regulares. Será explicada a metodologia utilizada, a arquitetura da aplicação e suas principais funcionalidades, com o propósito de demonstração da sua utilização e aplicação, traduzindo expressões regulares para linguagem natural restrita e também de linguagem natural restrita para expressão regular.
Palavras-chave: Regex. Expressões regulares. Padrão. Texto. Caracteres. Linguagem natural. Interpretação.

Title: Graduation Work Article – Translator Regex
Abstract: In this article will be presented to the translator Regex tool, created as a web application to facilitate interpretation and understanding of regular expressions. It will be explained the methodology used, the application architecture and its main functionalities, for the purpose of demonstration of its use and application, translating regular expression for restricted natural language and also restricted natural language for regular expression.
Key-words: Regex. Regular expressions. Pattern. Text. Characters. Natural language. Interpretation.

Introdução

Expressão regular. Algo que soa complexo e que muitos torcem o nariz ao ouvir falar. Mas afinal, o que são expressões regulares?

"Uma composição de símbolos, caracteres com funções especiais, que, agrupados entre si e com caracteres literais, formam uma sequência, uma expressão. Essa expressão é interpretada como uma regra, que indicará sucesso se uma entrada de dados qualquer casar com essa regra, ou seja, obedecer exatamente a todas as condições."

Aurelio Marinho Jargas

"Número de CPF inválido. Preencha Novamente".

Um acontecimento comum, que pode ocorrer com qualquer desatento. Você está a navegar pela internet, preenche algum formulário ou acessa algum serviço online e recebe um aviso de que os dados estão incorretos e que terá que refazê-lo.
A verificação das informações que um usuário fornece é parte fundamental de qualquer aplicação web, há vários motivos pra isso: A aplicação deve dar ao usuário uma experiência fácil, garantir o processamento da informação e impedir que conteúdos inválidos e perigosos sejam fornecidos.
A validação destes dados é feita comparando os dados que o usuário fornece

com algum padrão reconhecido, como o uso de expressões regulares que identificam uma porção de texto e é utilizada para criar regras com o objetivo de validar as informações, como por exemplo, um endereço de e-mail, uma senha alfanumérica ou numérica, o CPF ou CNPJ, um CEP, etc. [1].

Objetivos e Justificativa

- Desenvolver uma aplicação web capaz de facilitar o emprego de expressões regulares.
- Quanto mais específico o padrão de texto que se quer encontrar, mais complexa e de difícil entendimento para leigos e iniciantes fica a expressão.
- O Tradutor de Regex irá facilitar o entendimento das expressões, transformando-as em comandos equivalentes em português, facilitando na criação de expressões regulares mais complexas.

Metodologia

Para aderir um padrão, optou-se pela separação entre os dados (Model) e o layout (View). Desta forma, alterações feitas no layout não afetam a manipulação de dados, e estes poderão ser reorganizados sem alterar o layout.

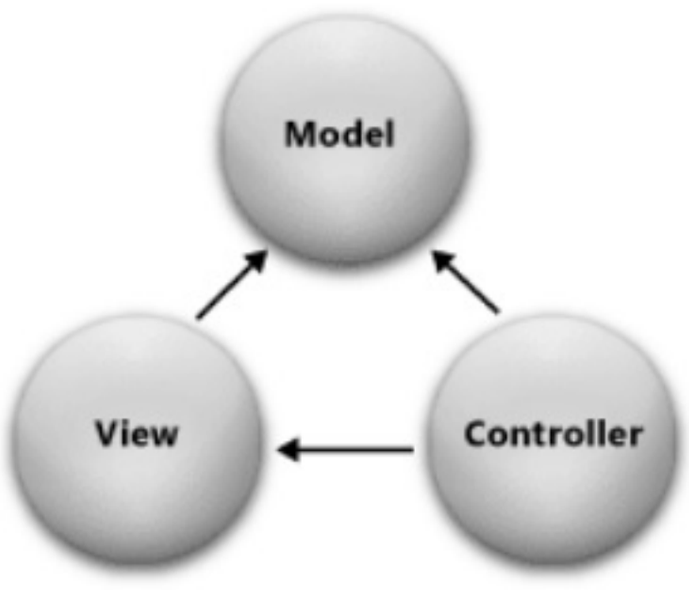

Figura 1: Modelo MVC (Model-View-Controller)

A arquitetura **MVC** - (Figura 1) fornece uma maneira de dividir a funcionalidade envolvida na manutenção e apresentação dos dados de uma aplicação.

Nesta arquitetura, o modelo representa os dados da aplicação e as regras do negócio que governam o acesso e a modificação dos dados. O modelo mantém o estado persistente do negócio e fornece ao controlador a capacidade de acessar as funcionalidades da aplicação encapsuladas pelo próprio modelo.

Um componente de visualização renderiza (processo pelo qual pode-se obter o produto final de um processamento digital qualquer) o conteúdo de uma parte particular do modelo e encaminha para o controlador as ações do usuário; acessa também os dados do modelo via controlador e define como esses dados devem ser apresentados.

Um controlador define o comportamento da aplicação, é ele que interpreta as ações do usuário e as mapeia para chamadas do modelo. Em um cliente de aplicações Web essas ações do usuário poderiam ser cliques de botões ou seleções de menus. As ações realizadas pelo modelo incluem ativar processos de negócio ou alterar o estado do modelo. Com base na ação do usuário e no resultado do processamento do modelo, o controlador seleciona uma visualização a ser exibida como parte da resposta a solicitação do usuário.

Camada de apresentação ou visualização (View representado na Figura 1) - Não está preocupada em como a informação foi obtida ou onde ela foi obtida apenas exibe a informação.

- Inclui os elementos de exibição no cliente: HTML e jsp.
- É a camada de interface com o usuário.

- É usada para receber a entrada de dados e apresentar o resultado

Camada de lógica da Aplicação (Model representado na Figura 1) - É o coração da aplicação. Responsável por tudo que a aplicação vai fazer.

- Modela os dados e o comportamento por atrás do processo de negócios: ANTLR[1]
- Se preocupa apenas com o armazenamento, manipulação e geração de dados.
- É um encapsulamento de dados e de comportamento independente da apresentação.

Camada de Controle (Controller representado na Figura 1)- determina o fluxo da apresentação servindo como uma camada intermediária entre a camada de apresentação e a lógica.

- Controla e mapeia as ações: Java Servlet. [2].

Desenvolvimento

Para a criação do Tradutor de Regex, utilizou-se alguns pacotes fundamentais para a tradução e exibição das expressões regulares.
A funcionalidade do sistema baseia-se no uso do ANTLR [3] que possibilita as traduções propostas. (Figura 2).

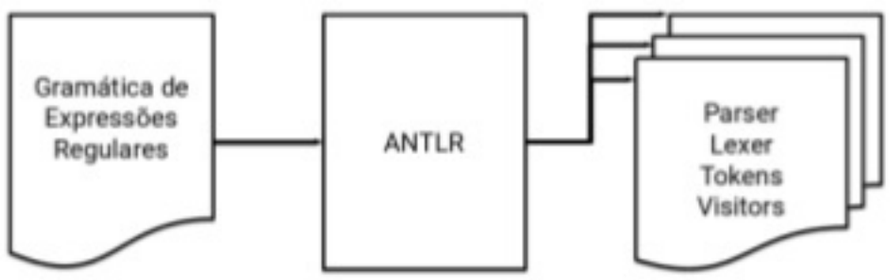

Figura 2: Exemplo da funcionalidade do sistema.

O ANTLR faz uso de regras léxicas e sintáticas para a tradução.

O Lexer (Exemplificado na Figura 3 – É feita a análise da entrada de caracteres e produção dos Tokens.) faz a análise léxica da linguagem de entrada, que é o processo de analisar a entrada de linhas de caracteres (tal como o código-fonte de um programa de computador) e produzir uma sequência de símbolos chamada "símbolos léxicos" (lexical tokens), ou somente "símbolos" (tokens) [4]. Identifica e classifica grupos de caracteres com um significado individual. Esse agrupamento é chamado *Tokenização*, e cada grupo identificado são chamados de Token, que serão enviados para o Parser.

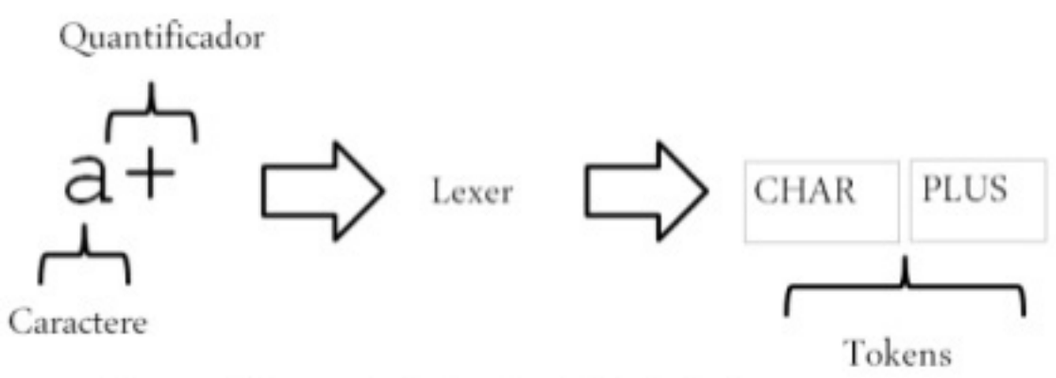

Figura 3: Exemplo da funcionalidade do Lexer.

O Parser faz a análise sintática da linguagem de entrada (Exemplificado na Figura 4 – Através da análise dos tokens, é definida sua estrutura gramatical, feita através de uma árvore de análise.), analisa uma sequência de entrada para determinar sua estrutura gramatical segundo a gramática formal [5]. Recebe os tokens gerados pelo Lexer e reconhece se a sintaxe da linguagem está correta.
Essa verificação é feita através de um parse tree (árvore de análise).

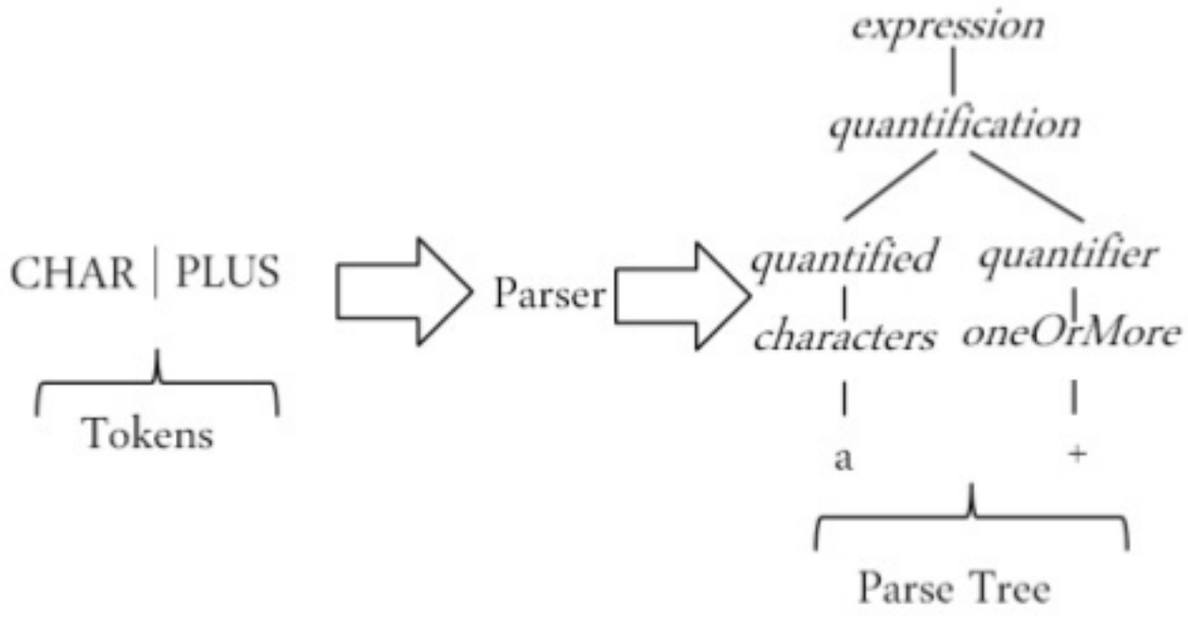

Figura 4: Exemplo da funcionalidade do Parser.

[1] **ANTLR:** (ANother Tool for Language Recognition) ANTLR converte a gramática que define uma determinada linguagem e gera todas as estruturas de dados e classes que implementam seu analisador léxico e o Parser desta nova linguagem.

O algoritmo de tradução precisa realizar operações para cada elemento existente da árvore de análise.
A interface *Visitor* provê um modo de "visitar" todos os elementos da árvore. (Figura 5).

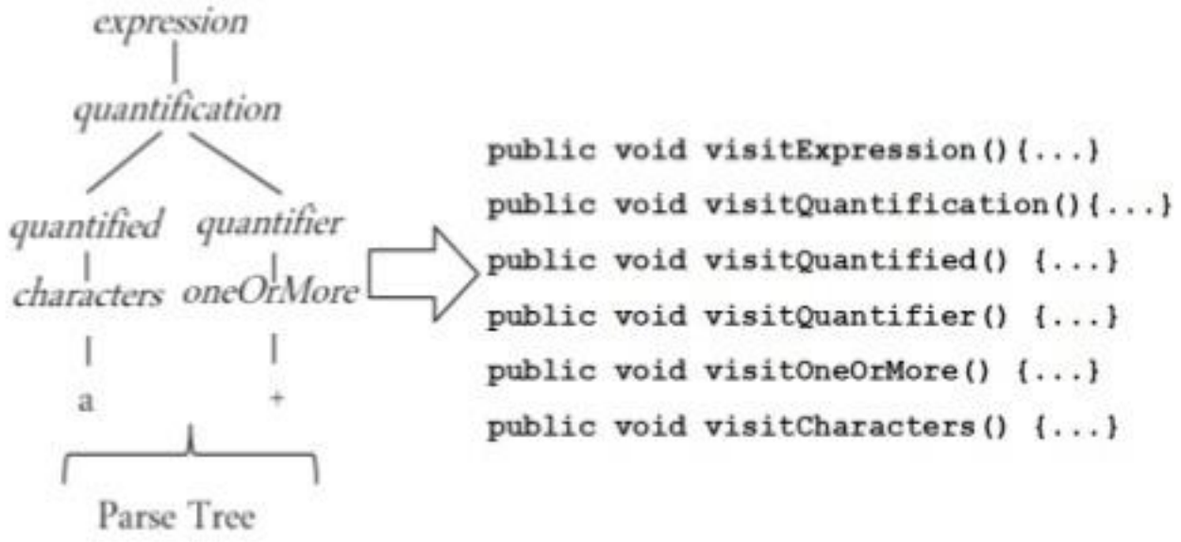

Figura 5: Exemplo da funcionalidade do Visitor.

Assim tem-se o site Regex:

Onde o usuário entra com os dados a serem traduzidos e, o sistema realiza a tradução (Exemplificação na Figura 6). Exibindo o resultado em linguagem natural restrita ou em expressão regular.

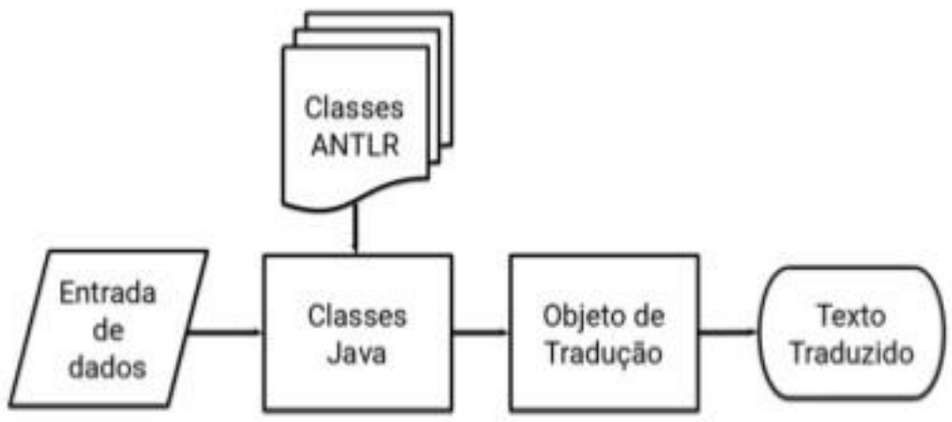

Figura 6: Exemplo da funcionalidade do sistema.

Resultados

Como resultado obtido no desenvolvimento da aplicação, o sistema executa as duas formas de tradução. Concluindo assim os dois incrementos propostos, tradução de expressão regular para linguagem natural restrita e tradução da linguagem natural restrita para expressão regular.
Tem-se como vantagem o layout da aplicação, que é dinâmico e de fácil entendimento. (Figura 7). A forma como a expressão é exibida, em árvore, facilita a compreender o que a expressão tende a fazer. Ainda não existe uma plataforma no mercado que realiza este tipo de tarefa sobre expressões regulares.

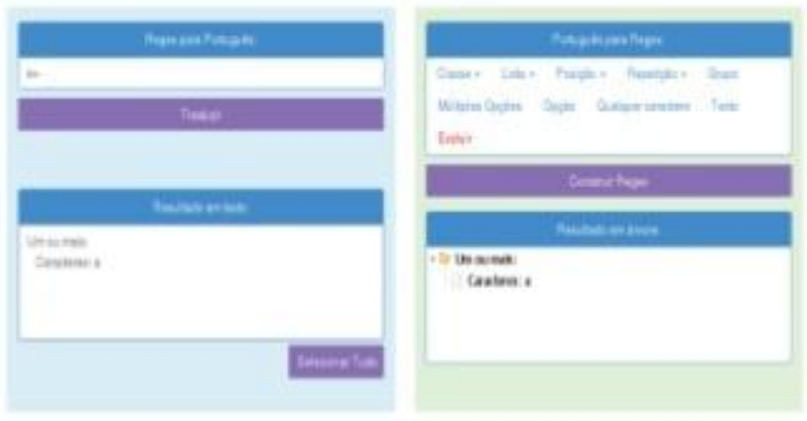

Figura 7: Protótipo de tela da aplicação.

Discussão e Conclusões

O projeto proporcionou conhecer novas ferramentas e ampliar nossos conhecimentos.
No decorrer do processo houve algumas dificuldades com relação à lógica e utilização dos pacotes que facilitariam a aplicação como um todo.
A biblioteca dhtmlxTree [6] deixou a desejar, dificultando no momento de adicionar um dado equivalente. Há utilização de um método específico da biblioteca que armazena esses dados em objetos Json [7] separados, complicando a utilização destes dados. Assim, a biblioteca força utilizar Json no projeto. Aceita outros tipos (Como por exemplo, XML), mas há várias restrições nestes casos.
Além disso, a biblioteca não possui muitas informações disponíveis a respeito de seu uso e funcionalidade.

Houve dificuldades também em escape de caracteres especiais [8].

Como trabalho futuro a aplicação poderá ser estendida e receber novas funcionalidades como a internacionalização da linguagem natural restrita, a mudança da sintaxe utilizada (Faz-se uso do padrão POSIX ERE 1003.1 [9]. Existem outras sintaxes disponíveis com padrões diferentes.) e a evolução da aplicação, desenvolvendo um *plugin* que dará suporte aos programadores.

Agradecimentos

Agradecemos aos professores Nelson Augusto Oliveira de Aguiar e Milkes Yone Alvarenga, por toda a colaboração com nosso projeto.
Ambos estiveram disponíveis em saciar nossas dúvidas e nos ajudar com a evolução da aplicação, deixando-a da melhor maneira possível.

Referências

[1] Opera House **"Expressões regulares".**
Disponível em:
<http://operahouse.com.br/?u=expressoes-regulares> Acesso em 03 set. 2015.

[2] José Carlos Macoratti **"Padrões de projeto"**
Disponível em:
< http://www.macoratti.net/vbn_mvc.htm>
Acesso em 22 Out. 2015.

[3] Terence Parr **"O que é ANTRL".**
Disponível em: < http://www.antlr.org/>
Acesso em 11 Ago. 2015.

[4] Wikiversity **"Análise Léxica".**
Disponível em:
<https://pt.m.wikiversity.org/wiki/Introdu%C3%A7%C3%A3o_%C3%A0_Teoria_dos_Compiladores/An%C3%A1lise_L%C3%A9xica>
Acesso em 09 Out. 2015

[5] Wikiversity **"Análise Sintática".**
Disponível em:
<https://pt.m.wikiversity.org/wiki/Introdu%C3%A7%C3%A3o_%C3%A0_Teoria_dos_Compiladores/An%C3%A1lise_Sint%C3%A1tica>
Acesso em 09 Out. 2015

[6] DHX **"dhtmlxTree"**.
Disponível em:
<http://docs.dhtmlx.com/tree__index.html>
Acesso em 22 Out. 2015

[7] Alexandre Gama **"Introdução Json"**.
Disponível em:
<http://www.devmedia.com.br/introducao-json/23166>
Acesso em 22 Out. 2015

[8] Java Documentação **"Escape de caracteres"**.
Disponível em:
<https://docs.oracle.com/javase/tutorial/java/data/characters.html>
Acesso em 22 Out. 2015

[9] Open Group **"Expressões regulares"**.
Disponível em:
<http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap09.html>
Acesso em 11 Ago. 2015

MARINHO, A.J. Expressões regulares, Uma abordagem divertida. 4 ed. São Paulo: Novatec, 2012. 224 p.