Resumo

Actualmente a utilização de linguagens de *markup* é recorrente no mundo da tecnologia, sendo o HTML a linguagem mais utilizada graças à sua utilização no mundo da Web. Tendo isso em conta é necessário que existam ferramentas capazes de escrever documentos bem formados de forma eficaz. No entanto, a abordagem mais utilizada, *template engines*, tem dois problemas principais: 1) não garante a geração de documentos bem formados, 2) não garante um bom desempenho, devido à utilização de ficheiros de texto como ficheiros de template.

Para resolver o primeiro problema propomos que um template HTML passe a ser definido como uma *first-class function*. Para isto é necessário criar uma linguagem específica de domínio para que estas funções possam manipular a linguagem HTML. O nosso objectivo principal é criar as ferramentas necessárias para gerar linguagens específicas de domínio com base no ficheiro de definição da linguagem, explícito num ficheiro XSD. A linguagem de domínio gerada deve também garantir que as restrições da respectiva linguagem são verificadas. Removendo os ficheiros textuais que definem templates minimizam-se também os problemas de desempenho introduzidos pelo carregamento de ficheiros de texto e reduzem-se o número de operações sobre `Strings`.

A minha proposta, chamada `xmlet`, inclui ferramentas que possibilitam: 1) a análise e a extração de informação de um ficheiro XSD, 2) a geração de classes e métodos que definem uma linguagem específica de domínio que reflete as regras presentes no ficheiro XSD, 3) a abstração da utilização da linguagem de domínio gerada, com a utilização do padrão Visitor. Para validar esta solução criaram-se linguagens de domínio não só para HTML como também para a linguagem utilizada para definir layouts visuais para Android e para a linguagem das expressões regulares.

Comparando a solução desenvolvida com soluções semelhantes, incluindo *template engines* e algumas soluções com inovações face à abordagem dos *template engines*, obtemos resultados favoráveis. Verificamos que a solução sugerida é a mais eficiente em todos os testes feitos. Estes resultados são importantes, especialmente considerando que apesar de ser a solução mais eficiente introduz também a verificação das restrições da linguagem utilizada tendo em conta a sua definição sintática.

Palavras-chave: XML, eXtensive Markup Language, XSD, eXtensive Markup Language Schema Definition, Geração Automática de Código, Interface Fluente, Linguage Específica de Domínio.