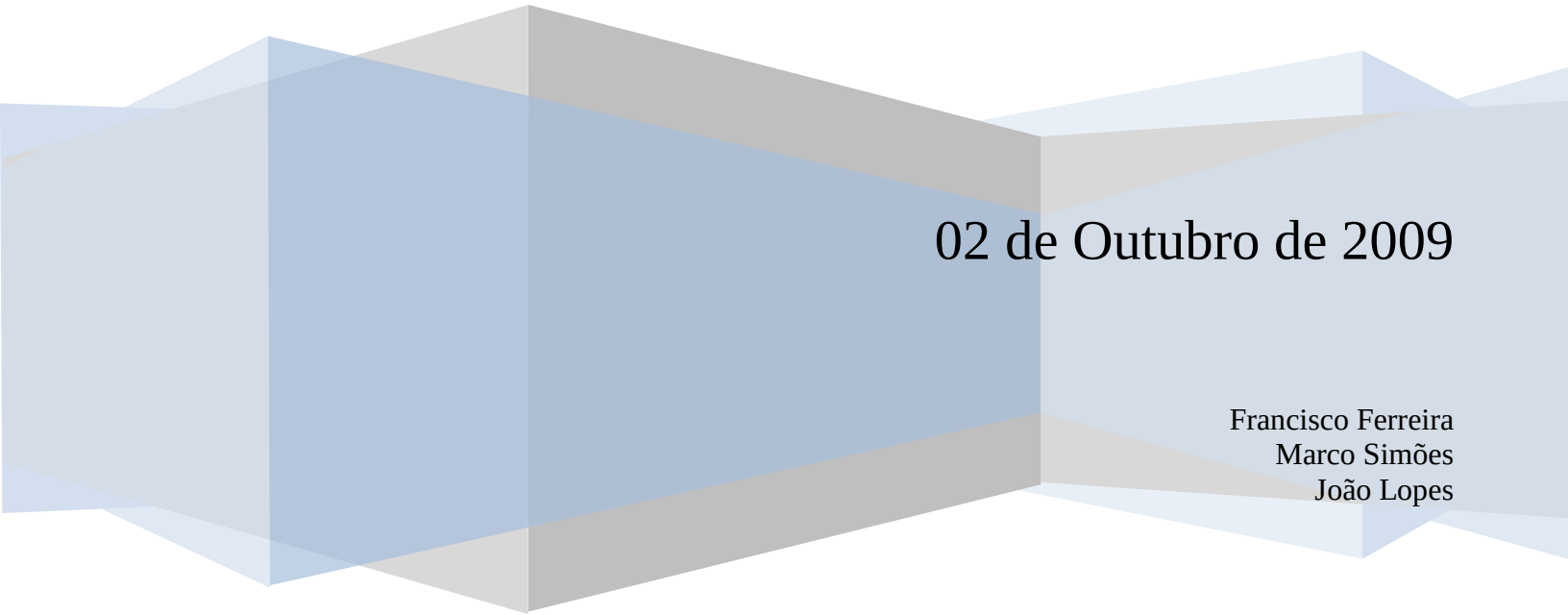
Integração de Sistemas

Relatório Projecto #1

Manipulação de XML



02 de Outubro de 2009

Francisco Ferreira
Marco Simões
João Lopes

1. Introdução

O XML tem um papel muito importante na Integração de Sistemas, sendo por isso vastamente utilizado em tecnologias tipo web-services e SOA (Service Oriented Architecture).

O objectivo deste trabalho é ambientar-nos com essa tecnologia. Mais especificamente construção, validação e uso de documentos no formato XML, com recurso a bibliotecas (como o JDom), XSD e XST/XSLT.

Para este projecto vai ser usado uma página de avaliações e características de câmaras fotográficas de várias marcas (http://www.dpreview.com/).

2. Componentes do projecto

O projecto está divido em 3 aplicações, cada uma com os seus objectivos distintos e trabalha nos ficheiros resultantes do aplicativo anterior.

O código fonte está organizado em 5 packages: is.apps, is.objects, is.parsing, is.util e is.xml. Na package apps encontram-se as MainClasses para as aplicações CameraSearchXML e CameraSummaryXML; Na package objects estão as classes que representam objectos, como por exemplo Camera ou Summary, bem como classes auxiliares a estas; Na package parsing estão as classes responsaveis por extrair a informação das páginas HTML através do recurso a expressões regulares; A package util emcapsula a class que usamos para fazer download das páginas HTML; E, por fim, a package XML contém a class que lê os ficheiros XML para extração de dados na 2ª aplicação, bem como uma class que serve de interface entre a aplicação e os ficheiro XML no disco, sendo esta responsável apenas por leituras e escritas.

Existem ainda as pastas conf, onde é colocado um ficheiro de configuração com as expressões regulares a utilizar e o endereço base do site, e a pasta xml, onde são por defeito colocados os ficheiros de output das aplicações. Esta última conté ainda os ficheiros necessários à 3ª aplicação (CameraListBeautifier) e os ficheiros de validação da estrutura dos XML.

2.1 CameraSearchXML

A primeira aplicação serve para descarregar o código fonte da página web e obter informação das câmaras de uma determinada marca (Canon, Nikon...), fornecida à applicação como argumento. Esta obtém os detalhes sobre cada câmara, usando para isso expressões regulares, e coloca-os dados em memória.

Após ter obtido os dados é gerado o XML com os dados da câmara e posteriormente criado um documento com todas as câmaras de uma marca. Tendo assim um ficheiro XML por cada marca de câmaras.

Foi criado também um XSD para validar os dados e a estrutura do XML gerado pelo programa.

2.1.1 "Screen Scrapping" e expressões regulares

Para realizar o "Screen Scrapping" foi necessário dividir internamente o parsing dos dados em duas fases: A primeira é responsável por extrair informação da primeira página, onde aparecem todas as câmaras listadas, nomeadamente o nome, o link para a página da câmara, a descrição e a data de anúncio para o mercado; A segunda é responsável por extraír a informação de cada máquina, que se encontra apenas na 2ª página.

A principal diferença entre as duas fases é que a primeira é feita por tópico e a segunda por máquina. Isto é, na primeira fase temos que extrair primeiro o nome de todas a máquinas, depois as descrições de todas, etc. Na segunda fase, entramos na página da máquina e extraímos a informação respectiva a cada componente dessa máquina.

Relativamente às expressões regulares, estas foram deixadas independentes, isto é, cada campo de informação é extraído por uma expressão regular, não havendo uma expressão regular responsável por retirar informação de mais que um componente da máquina. Isto permite maior tolerância a alterações da página original, evitando o efeito cascata na propagação de um erro. Além disso, estas foram deixadas configuráveis, de modo a que possam ser alteradas no ficheiro de configuração, sem ter que andar a alterar o código do projecto.

2.1.2 Estrutura do XML e XSD

Vendo os dados oferecidos pelo web-site criámos e reformulámos a estrutura XML até chegarmos à nossa versão actual, que serve os seus propósitos neste trabalho. Tentámos hierarquizar de forma correcta os campos que parecessem relacionados entre si, como por exemplos as resoluções que tem 3 níveis de profundidade. A estrutura final está definida como é mostrado na imagem seguinte:

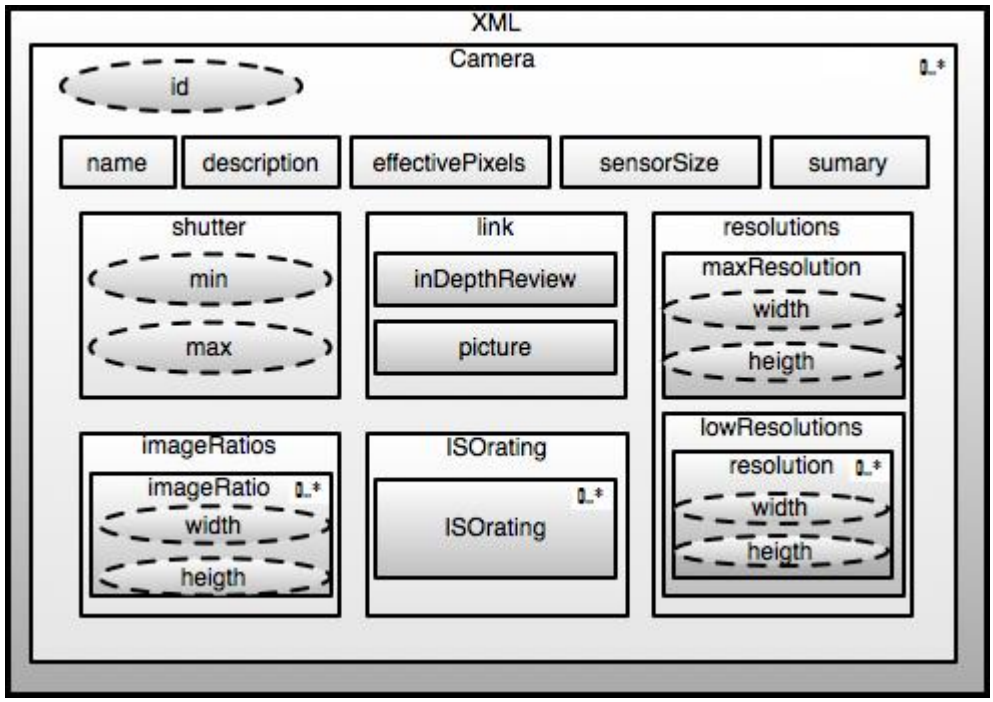


Fig. 1

Na Fig. 1 é possível ver a estrutura básica do XML gerado pela aplicação. Os quadrados representam elementos e os círculos atributos. Está também representada algumas regras de validação. "0..*" quer dizer que esse elemento pode não existir ou existir vários elementos. Todos os outros elementos existem 1 só vez.

A separação entre atributos e campos do XML foi feita de modo a evitar ao máximo redundância de dados e uma estrutura lógica e correcta. Que foi uma preferência escolhida para as resoluções que em vez de ter elementos "width" e "heigth", temos várias resoluções mas os dados deste são atributos. Assim foi aproveitada a estrutura e gramática do XML de modo a se escrever um documento mais bem definido, transparente e simples.

Nos casos de inexistência de informação respectiva a um campo, optámos por colocar a tag xml vazia, mostrando que a informação seria mostrada, mas não estava disponível. A simples eliminação da tag deixaria sempre a possíbilidade de haver a informação, mas nao ter sido mostrada devido a um erro da aplicação. Desta forma, fica transparente para o destinatário que o problema está na fonte da informação.

O ficheiro XSD valida a estrutura apresentada na imagem, mas num nível mais baixo. Isto é, este obriga a que os campos numéricos sejam números, as datas sejam datas, os textos sejam textos, etc. Como as tag's são sempre colocadas, mesmo que não haja informação, a validação fica mais simplificada, podendo obrigar sempre a existência das tags.

2.2 CameraSummaryXML

Esta aplicação vai aceder aos XML gerados pelo aplicativo anterior e seleccionar algumas câmaras com atributos especiais de cada marca, tais como a mais recente e a que tem maior resolução. Faz esta selecção para cada marca.

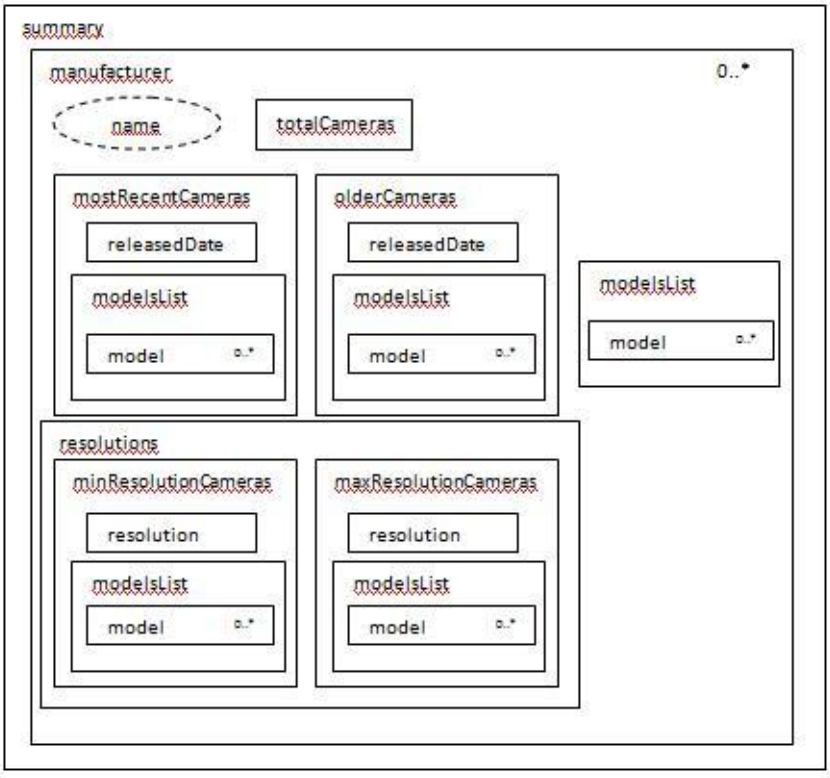
Resulta então um XML com as câmaras associadas à característica que as seleccionou para estarem no ficheiro. Como podem ser fornecidos diversos ficheiros como input, o XML final está separado por manufacturers, indicando para cada o respectivo sumário.

2.2.1 Parsing do XML

Para leitura dos ficheiros XML utilizamos a biblioteca jdom que, no caso de XML bem formados, cria uma extrutura em memória dos dados, facilitando em larga escala o acesso e transformação dos mesmos. Desta forma torna-se relativamente fácil descobrir quais as máquinas com melhor resolução, as mais recentemente lançadas, entre outras.

2.2.2 Estrutura do XML e XSD

Nesta fase deparámo-nos com um pequeno problema. Pederiam existir várias máquinas com a melhor resolução, por exemplo. Desta forma, criámos uma estrutura que nos permitisse apresentar vários modelos que apresentassem a mesma característica.



Como se pode ver na Fig. 2, as minResolutionCameras apresentam a resolução e depois uma lista com os modelos que têm essa resolução.

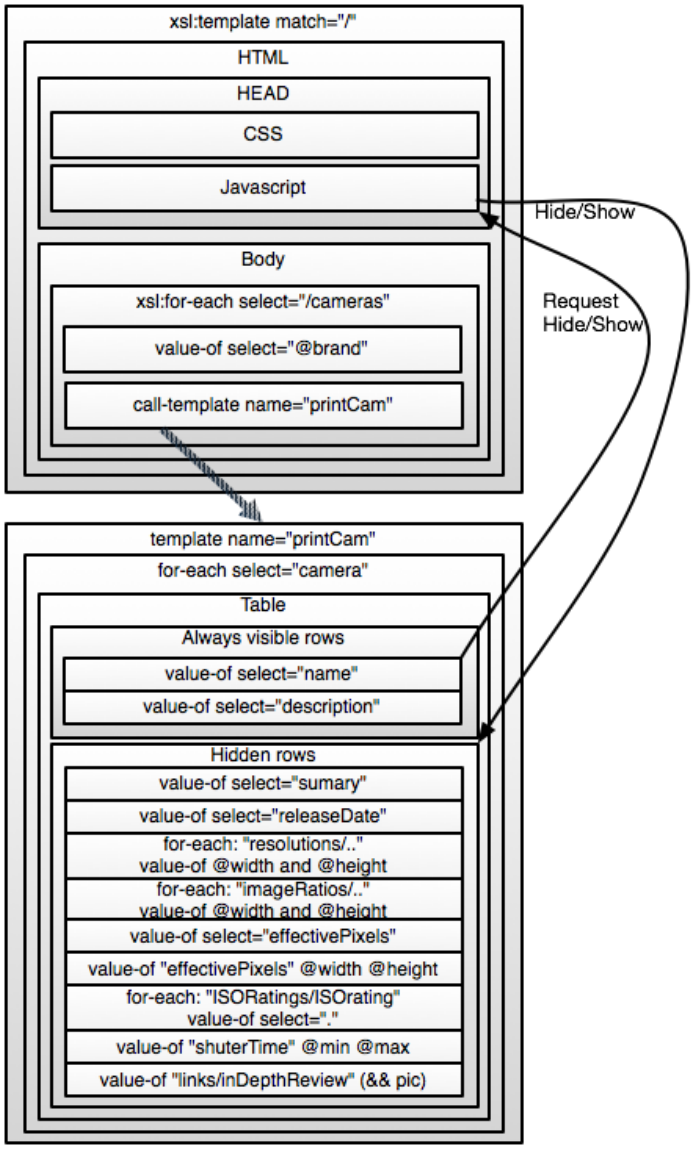
Quanto ao XSD, assemelha-se ao da aplicação anterior, adaptando-se à estrutura deste xml. Mantém-se o mesmo rigor associado ao tipo de dados, por exemplo.

2.3 CameraListBeautifier

O beutifier consiste num ficheiro XSL que, ao ser incorporado no XML, dá os recursos necessários para um browser, com suporte de XSLT, apresentar os dados do XML de um forma mais legível e facilmente consultada. O motor XLST lê o ficheiro XML e usando as regras no XSL constrói um ficheiro HTML com os dados do XML.

2.3.1 Estrutura do XSL

O XSL foi escrito em conjunto com javascript e CSS de modo a, para além de dar uma estrutura mais user-friendly ao XML, também lhe proporcionar algum dinamismo e interactividade com o utilizador. Isto fez com que se tenha evoluído um pouco a estrutura inicialmente implementada para o HTML, tendo-se assim adicionado às tags <tr> atributos de class e de id dinamicamente, a partir do XML, em vez de se poder adicionar as classes directamente ao código gerado. O XSL tem a seguinte estrutura:



3. Ant script

Segue com o projecto um ficheiro build.xml, que permite a um motor ant compilar e executar a aplicação. Desta forma, para executar a aplicação basta encontrar-se na pasta do projecto e executar os seguintes comandos:

CameraSearchXML:

ant main1 -Dargs="Canon" //para executar a aplicação com a marca Canon

CameraSummaryXML:

ant main2 -Dargs="xml\Canon.xml xml\Casio.xml xml\Agfa.xml" //para executar a aplicação com 3 ficheiros referidos

3. Tempo dispendido para tarefas:

Como requesitado no enunciado do trabalho, aqui se encontram os tempos despendidos de cada elemento por tarefa. Há tempo comum a todos os elementos, que é de cerca de 3 horas, que se deve a "reuniões" para decidir os caminhos e as metodologias a tomar pelo projecto.

O tempo de leitura de tutorials não foi incluído na totalidade. Algum tempo está incluído na implementação e outro não foi medido.

3.1 Francisco Ferreira

16 horas a implementar o XSL + Javascript + Html + CSS
5 horas no relatório.

3.2 João Lopes

14 horas na implementação da obtenção do código fonte da página + XSD + XMLSearch
7 horas no relatório

3.3 Marco Simões

17 horas na implementação das expressões regulares + XSD + XMLSummary
3 horas no relatório

02 de Outubro de 2009

Francisco Ferreira
fmsf@student.dei.uc.pt
2006124182

João Lopes
jmlopes@student.dei.uc.pt
2006125131

Marco Simões
msimoes@student.dei.uc.pt
2006125287

Integração de Sistemas
Mestrado em Engenharia Informática no
Departamento de Engenharia Informática da
Faculdade de Ciências e Tecnologia da Universidade de Coimbra