COBENGE 2014
ENGENHARIA: Múltiplos saberes e atuações
16 a 19 de setembro | Juiz de Fora - MG

APLICAÇÃO DO ENSINO POR PESQUISA NO DESENVOLVIMENTO DE WEB SERVICE PARA MANIPULAÇÃO DE ARQUIVOS UTILIZANDO JAVA SERVLET PAGES.

Fredericko Xavier Gonçalves - fredericko.xg@hotmail.com
Geshner Inagaki Martins - geshner.inagaki@gmail.com
Ruy de Oliveira - ruy@cba.ifmt.edu.br
Ed' Wilson Tavares Ferreira – edwilson.ferreira@ifmt.edu.br
Valtemir Emerencio do Nascimento – valtemir.nascimento@cba.ifmt.edu.br
IFMT/ Departamento de Informática
Rua Professora Zulmira Canavarros, 95, Centro
78005-200 – Cuiabá – Mato Grosso

***Resumo*:** *Com o avanço da tecnologia web os serviços também mudaram de simples páginas estáticas para páginas dinâmicas. Este artigo apresenta o uso da tecnologia Java Server Pages para o desenvolvimento de um website dinâmico capaz de realizar upload e compactação de arquivos diversos e manipulação de arquivos em PDF, por meio de levantamento bibliográfico. O desenvolvimento desta página foi proposto em sala com o intuito de colocar em prática o método de aprendizagem por pesquisa, com o fim de verificar os resultados obtidos por parte dos discentes.*

Palavras-chaves: *Servlet, JSP, Java Server Pages*

1. INTRODUÇÃO

A tecnologia de serviços web traz consideráveis benefícios para o desenvolvimento de aplicações. A principal vantagem do uso dos serviços web é a interoperabilidade, obtida através de protocolos web, sendo um deles o Protocolo de Transferência de Hipertexto (HTTP), o qual permite a navegação na internet através de hiperlinks.

Atualmente, grande parte do conteúdo que é acessado na internet, como portais de notícias, blogs, home banking, entre outros são baseados em conteúdo dinâmico. Diferentemente de páginas estáticas que se popularizaram nos primórdios da internet, hoje o usuário requisita algo ao servidor, e este processa a requisição e devolve uma nova forma de resposta ao usuário. Essa interação tornou-se possível devido à consolidação dos serviços oferecidos na web, baseada em iniciativas de serviços disponibilizados de forma *on-line*, entre os quais destacam-se serviços de busca/pesquisas e serviços governamentais.

Tais serviços na web, possibilitaram a construção de aplicações distribuídas e colaborativas, em que os serviços oferecidos pelas aplicações estão na forma de módulos distribuídos na rede global, sendo assim localizados e invocados dinamicamente nos moldes da arquitetura

orientada a serviços.

Este artigo está dividido em três partes. Na primeira parte apresenta-se os principais conceitos envolvidos na tecnologia de serviços web utilizadas, e na segunda parte descreve-se o desenvolvimento de um *website* dinâmico que oferece serviços como a compactação de arquivos e a manipulação de arquivos no formato PDF, fazendo uso da tecnologia *Java Servlet Pages* (JSP) e *Servlet*, conforme proposta feita pelos professores envolvidos com a disciplina de Tecnologias Cliente-Servidor. Nesta disciplina, ministrada no quarto semestre da graduação do curso de tecnologia em Sistemas para a Internet, aborda-se assuntos relacionados com as tecnologias e serviços utilizados na comunicação entre as máquinas cliente e servidor. Na última parte do artigo os resultados obtidos são apresentados, assim como as dificuldades e sugestões dos discentes.

2. FUNDAMENTAÇÃO TEÓRICA

Potts e Kopack (2003) define *web service* como uma aplicação de software que pode ser acessada remotamente através de diversas máquinas distintas. Usando linguagens baseadas em XML (Extensible Markup Language) um *webservice* é identificado pelo seu endereço URL, como qualquer *website*, porém o que o distingue de um *website* comum é o tipo de informação que o webservice pode fornecer ao usuário. O *website*, é projetado de forma que possa fornecer uma resposta para a requisição feita pelo usuário, quando este digita um endereço URL, ou clica em um *hyperlink*, fazendo com que o servidor tenha apenas que retornar um documento. A figura abaixo ilustra este processo.

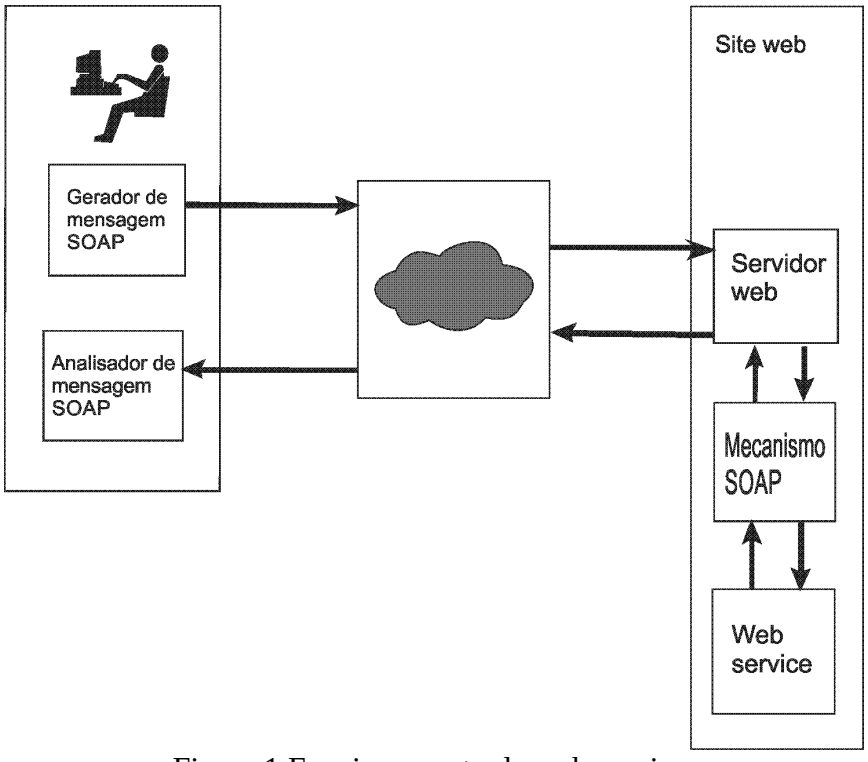

Figura 1 Funcionamento do web service.

Já no *webservice* além de receber as requisições por endereços URL e *hyperlinks*, os dados que são enviados ao servidor são diferentes dos dados enviados através do *website*, os clientes que utilizam o *webservice* enviam um documento no formato XML para o servidor conforme as regras de especificações SOAP (Simple Object Access Protocol). Este processo está ilustrado na imagem a baixo.

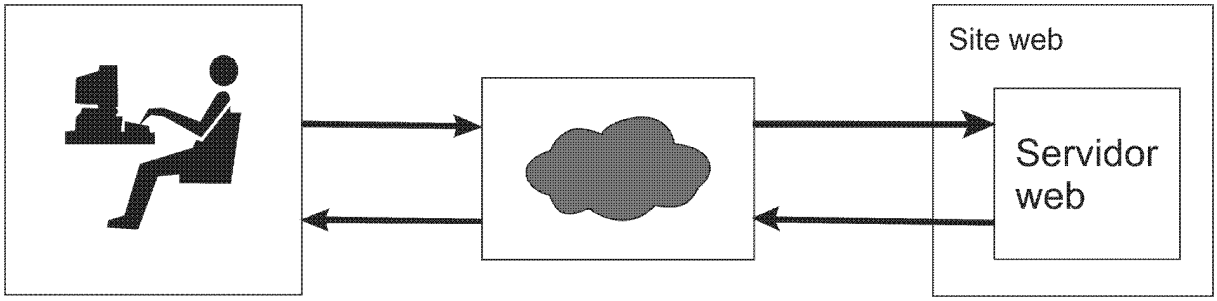

Figura 2 Comunicação entre cliente e servidor utilizando SOAP.

Uma das vantagens de se utilizar o *webservice* está na interoperabilidade, visto que a arquitetura desta tecnologia está baseada no envio de arquivos com formatações específicas, facilitando assim o uso da mesma informação em computadores com configurações distintas.

Java Server Pages (JSP) é uma tecnologia gratuita desenvolvida pela empresa *Sun microsystems*, que possui como objetivo primário a geração de conteúdo dinâmico para páginas da internet. Como o foco desta tecnologia é o dinamismo na web pode-se mesclar os códigos da Linguagem de Marcação de Hyper-Texto (HTML) que geram conteúdos estáticos, com os códigos JSP para se criar o dinamismo que se deseja.

Por ser uma tecnologia gratuita, a JSP possui especificações abertas. Por isso, é possível configurar um servidor que dê suporte a esta tecnologia, e entre estes se destacam os servidores, TomCat, GlassFish e JBoss.

Como a JSP é uma linguagem *Server-side script* (script executado do lado do servidor), o usuário não dispõe de acesso ao seu código fonte, já que este é convertido pelo servidor. Então, o usuário final recebe apenas o código HTML . (ANSELMO, 2005).

De acordo com a ORACLE, os *Servlets* representam uma classe da linguagem de programação JAVA, que herda as funções do servidor que hospeda a aplicação *web*, e a principal função herdada é a função de manipulação de requisições do tipo HTTP. Dessa forma, o *servlet* é um componente que realiza o tratamento de todos as requisições enviadas pelo cliente ao servidor, informando ao servidor qual método deve ser executado ou não e o que deve ser enviado como resposta ao cliente.

Neste trabalho optou-se por usar o servidor Apache Tomcat versão 7, recomendado pelo professor que ministrou diretamente a disciplina de Tecnologias de Cliente-Servidor, e para a codificação Java no cliente e no servidor utilizou-se o Ambiente de Desenvolvimento Integrado (IDE) Eclipse Kepler.

3. METODOLOGIA

O desenvolvimento do *website* foi feito com base em levantamento bibliográfico feito por parte dos alunos. Apesar da entidade mantedora da tecnologia JSP ter descontinuado a tecnologia ainda há muitas aplicações no mercado que ainda a utilizam. Por haver uma grande

quantidade de aplicações feitas com esta tecnologia optou-se por utilizá-la. Esperava-se assim facilidade na obtenção de material didático acerca do assunto, bem como na obtenção de aplicações a fim de viabilizar a análise e o aprendizado por parte dos estudantes envolvidos. Apresenta-se a seguir como foi desenvolvido cada método proposto neste trabalho.

3.1. Upload de arquivos

Para o *upload* de arquivos, primeiramente foi desenvolvido um formulário HTML em um arquivo JSP, e no formulário é possível escolher os arquivos que serão enviados ao servidor. Para que o usuário do sistema possa escolher qualquer arquivo de sua máquina, adicionou-se um *input* do tipo *file* possibilitando assim a seleção de qualquer arquivo. Foi acrescentado para este campo o atributo "*multiple*", o qual permite a seleção de múltiplos arquivos. O procedimento de envio de arquivo ao servidor só é iniciado quando o usuário efetua um *click* no botão de envio disponível no formulário.

Ao iniciar o serviço de *upload* do arquivo, o *servlet* responsável por este procedimento é chamado, dando início a verificação do tipo de transferência para determinar se a transferência é do tipo *multipart*, ou seja transmissão em múltiplas partes. Somente depois dessa verificação é que o arquivo pode ser gravado no diretório pré-definido no arquivo *web.xml*. O conteúdo deste arquivo determina as características do arquivo a ser gravado, tais como os diretórios que a aplicação possui, quais *servlets* estão disponíveis e em quais locais da pasta no servidor. O upload em si é feito utilizando-se as classes *ServletUpload*, que guarda todos os itens em um vetor de *bytes*, enquanto o FileItemIterator guarda apenas um item por vez. Para cada arquivo recebido do cliente é aberto seu InputStream, o que possibilita o acesso aos bytes do arquivo, por meio do método copy da classe Streams, o qual escreve o arquivo no diretório, conforme mostrado na Figura 1.

Figura 3 – Servlet de Upload

```
boolean isMultipart = ServletFileUpload.isMultipartContent(request);
if (isMultipart) {
    ServletFileUpload upload = new ServletFileUpload();
    FileItemIterator iter;
    try {
        iter = upload.getItemIterator(request);
        while (iter.hasNext()) {
            FileItemStream item = iter.next();
            InputStream stream = item.openStream();
            if (item.isFormField()) {
            } else {
                File file2 = new File(getServletContext().getInitParameter("diretorio")
                        +new File(item.getName()).getName());
                FileOutputStream fos = new FileOutputStream(file2);
                Streams.copy(stream, fos, true);
            }
        }
```

Fonte: Autores.

3.2 Compactação de arquivos

Para este serviço também foi utilizado um formulário feito em HTML com um campo

para seleção de arquivos, um *input* do tipo *file* com atributo *multiple*, possibilitando a seleção de múltiplos arquivos para o envio e compactação. Este formulário possui juntamente ao campo de escolha de arquivo um botão para o envio dos arquivos selecionados pelo usuário. Quando o usuário pressiona o botão é feito o envio dos arquivos ao servidor da mesma maneira que é feita no método de upload, porém o *servlet* responsável por este procedimento é diferente do servlet utilizado na proposta anterior.

O servlet de compactação de arquivos ao receber a solicitação, verifica o tipo de transmissão de dados, e caso seja uma transmissão em múltiplas partes o mesmo realiza o upload dos arquivos para uma pasta pré-definida no servidor. Em seguida, a compactação é iniciada utilizando o ZipOutputStream, uma classe da linguagem de programação JAVA que permite a criação de arquivos no formato zip, ou seja compactado.

A compactação em si ocorre da seguinte maneira. Primeiro cria-se uma variável com um endereço e nome "arquivo.zip" que será o arquivo compactado gerado pelo servidor. Depois disso executa-se um laço de repetição para cada arquivo existente na pasta que contém os arquivos que foram escolhidos pelo cliente para compactação. Na sequência, executa-se o comando *write* da classe *ZipOutputStream* para a escrita do arquivo compactado.

No final do procedimento de compactação o arquivo é gerado e enviado ao cliente solicitante da operação de forma que o seu download comece de forma imediata. Na Figura 2 pode-se conferir o método de compactação de arquivo.

Figura 4 – Servlet de Compactação

```
boolean isMultipart = ServletFileUpload.isMultipartContent(request);
if (isMultipart) {
    ServletFileUpload upload = new ServletFileUpload();
    FileItemIterator iter;
    try {
        iter = upload.getItemIterator(request);
        File fileZip = new File(getServletContext().getInitParameter("diretorio"), "arquivo.zip");
        ZipOutputStream zos = null;
        FileOutputStream fos2 = null;
        fos2 = new FileOutputStream(fileZip);
        zos = new ZipOutputStream(fos2);
        FileInputStream fis;

        while (iter.hasNext()) {
            FileItemStream item = iter.next();
            InputStream stream = item.openStream();
            if (item.isFormField()) {
            } else {
                String nome = new File(item.getName()).getName();
                File file2 = new File(getServletContext().getInitParameter("diretorio"), nome);
                FileOutputStream fos = new FileOutputStream(file2);
                Streams.copy(stream, fos, true);

                byte[] arqZip = new byte[1024];

                fis = new FileInputStream(file2);
                zos.putNextEntry(new ZipEntry(nome));

                int len;
                while ((len = fis.read(arqZip))> 0) {
                    zos.write(arqZip, 0, len);
                }
                zos.closeEntry();
                fis.close();
                fos.close();
            }
        }
        zos.close();
        fos2.close();
```

Fonte: Autores.

4. CONSIDERAÇÕES FINAIS

Este artigo apresentou o desenvolvimento de uma aplicação web em jsp didática para prover serviços de upload e compactação de arquivos por meio do modelo Cliente-Servidor. A aplicação foi criada por estudantes de graduação como uma forma de estímulo ao aprendizado por meio da investigação científica.

A aprendizagem com o método de ensino empregado possibilitou aos estudantes experimentar as dificuldades de um ambiente real de desenvolvimento de aplicativos similares ao que foi proposto neste trabalho, bem como a conscientização desses estudantes da necessidade de se buscar a superação de limites. O docente que ministrou a disciplina observou uma nítida evolução da habilidade dos estudantes na resolução dos problemas propostos, o que certamente se deveu a autonomia de aprendizado conquistada por esses discentes.

Como melhorias para a prática que foi empregada no semestre do desenvolvimento deste trabalho, os estudantes sugerem: mais momentos de troca de experiência com o professor que trabalhou diretamente com os estudantes, mais exemplos em sala de aula que possibilitem aos estudantes obter mais embasamento acerca do assunto ensinado e material

bem atualizado sobre as novas tecnologias pertinentes.

REFERÊNCIAS BIBLIOGRÁFICAS

ANSELMO, Fernando. **Tudo sobre a JSP – com o NetBeans em Aplicações Distribuidas**, Visual Books, 2005.

CRUZ, Sérgio Manuel da. **Serviços Web – Uma breve introdução (Parte I)**. Disponível em: < http://www.nce.ufrj.br/conceito/artigos/2005/01p1-1.htm >. Acesso em: 16 abr. 2014.

ROCHA, C. A. **Desenvolvendo Web Sites Dinâmicos - PHP, ASP, JSP**, Campus, 2003.

SIERRA, Kathy. **Use a Cabeça! Servlets & JSP**, Alta Books, 2008.

POTTS, Stephen; KOPACK, Mike. **Aprenda em 24 horas Web services**. Rio de Janeiro, Campus, 2006.

APPLICATION OF THE INQUIRY BASED LEARNING IN DEVELOPING WEB SERVICES FOR ARCHIVES MANIPULATION USING JAVA SERVLET PAGES.

Abstract: *With the progress of web technology, web services have also changed from simple static pages to dynamic pages. This paper presents the use of Java Server Pages technology to develop a dynamic website able to upload and compress archives and the manipulation of PDF files through a literature review. The development of this page was proposed during class with the intention of putting into practice the method of Inquiry-based learning to verify the results obtained by students.*

Keywords: *Servlet, JSP, Java Servlet Pages*