Documentos

Setembro, 2003 28

ISSN 1677-9274

Publicando Mapas na Web: Servlets, Applets ou CGI?

Embrapa

ISSN 1677-9274
Setembro, 2003

Empresa Brasileira de Pesquisa Agropecuária
Embrapa Informática Agropecuária
Ministério da Agricultura, Pecuária e Abastecimento

Documentos 28

Publicando Mapas na Web: Servlets, Applets ou CGI?

José Iguelmar Miranda

Campinas, SP
2003

Embrapa Informática Agropecuária
Área de Comunicação e Negócios (ACN)
Av. André Tosello, 209
Cidade Universitária "Zeferino Vaz" – Barão Geraldo
Caixa Postal 6041
13083-970 – Campinas, SP
Telefone (19) 3789-5743 - Fax (19) 3289-9594
URL: http://www.cnptia.embrapa.br
e-mail: sac@cnptia.embrapa.br

Supervisor editorial: *Ivanilde Dispato*
Normalização bibliográfica: *Marcia Izabel Fugisawa Souza*
Capa: *Intermídia Produções Gráficas*
Editoração eletrônica: *Intermídia Produções Gráficas*

1ª. edição on-line - 2003

Miranda, José Iguelmar.
Publicando mapas na web: servlets, applets ou cgi? / José Iguelmar Miranda. – Campinas : Embrapa Informática Agropecuária, 2003.
38 p. : il. – (Documentos / Embrapa Informática Agropecuária ; 28)

ISSN 1677-9274

1. Publicação de mapas. Mapas na Internet. 3. Criação de mapas. 4. Uso de mapas. 5. Servlets. 6. Applets. 7. CGI. I.Título. II. Série.

CDD – 005.720223 21st ed.

Autor

José Iguelmar Miranda
Ph.D. em Geoprocessamento, Pesquisador da Embrapa Informática Agropecuária, Caixa Postal 6041, Barão Geraldo
13083-970 - Campinas, SP.
Telefone (19) 3789-5736 – e-mail: miranda@cnptia.embrapa.br

Apresentação

Existe uma grande oferta de aplicativos para publicar mapas pela Web. Este trabalho tem como objetivo analisar três deles dando ênfase nas suas arquiteturas de implementação, procurando mostrar ao usuário os requisitos, vantagens e desvantagens de cada tipo de arquitetura no ambiente da Internet.

Estas arquiteturas classificam-se em: *applets*, *servlets* ou aplicativos usando *Common Gateway Interface* (CGI). *Applets* e *servlets* são tecnologias Java, da Sun Microsystems, e estão se tornando progressivamente um padrão para desenvolver aplicações na Web. *Applets* são aplicações do tipo cliente que permitem interatividade do usuário com a aplicação, sendo indicadas para produzir interfaces gráficas. *Servlets* são aplicações do tipo servidor, porém não permitem tanta interatividade como os *applets* nem são indicadas para interfaces gráficas. *Common Gateway Interface* não é uma linguagem de programação, mas uma maneira de uma linguagem de programação "tradicional", como C/C++ ou uma linguagem do tipo *script* (interpretada), como PERL ou PHP, se comunicar com o padrão *Hyper Text Transfer Protocol* (HTTP). Aplicativos CGI foram identificados como as primeiras tentativas de desenvolver aplicações genéricas para a Web. Estes aplicativos, à semelhança dos *servlets*, são aplicações do tipo servidor.

A quantidade de aplicativos disponíveis para publicar mapas na Web, baseada em alguma destas tecnologias, só tem crescido nos últimos cinco anos. Como exemplo, os serviços de localização (ruas, estradas, lojas, etc.), disponíveis pela Internet. Nos celulares e nos computadores de bordo dos carros mais modernos, serviços de localização também estão disponíveis usando a tecnologia WAP (*Wireless Application Protocol*) (Wirelless Application Protocol Forum Ltd., 2002). Existem instituições, com e sem fins lucrativos, que disponibilizam tais aplicativos. Surge então uma pergunta: qual tipo de aplicação seria melhor usar? Se é que pode haver tal alternativa. O que se deve saber para basear a escolha? Neste trabalho, estas e outras questões relacionadas serão apresentadas com o objetivo de ajudar o interessado na tomada de decisão.

José Gilberto Jardine
Chefe-Geral

Sumário

Publicando Mapas na Web: Servlets, Applets ou CGI?

José Iguelmar Miranda

Introdução

O objetivo deste trabalho é fornecer subsídios para se escolher entre três tipos de aplicativos para publicação de mapas pela Web. Programas para a Internet classificam-se em dois grupos: Java ou não Java. A linguagem Java foi desenvolvida pela Sun Microsystems, com o objetivo de ser uma linguagem de programação independente do sistema operacional do computador, com fundamentos de orientação a objetos, visando segurança na rede e facilidade para substituir código executável nativo. No grupo Java as aplicações mais comuns são do tipo *applet* ou *servlet*. No grupo não Java tem-se as linguagens de programação compiladas ou as linguagens interpretadas. Ambas opções fazem uso de um protocolo de comunicação com a Internet, o *Common Gateway Interface* (CGI). Neste trabalho, adota-se a nomenclatura aplicativo CGI a esta combinação linguagem/protocolo apenas para facilitar a identificação. Portanto, sistemas para publicar mapas na Web, no fim, pertencem a uma de três categorias: *applet*, *servlet*, ou aplicativo CGI.

Um mundo digital, onde se pode encontrar todo tipo de informação em mídia eletrônica, tem se tornado uma realidade. Ninguém pode negar que a cada dia centenas de milhares de textos, figuras, sons e imagens são disponibilizadas na Internet. E isto não é diferente para mapas. Pode-se encontrar na Web mapas com diferentes acabamentos, ou precisão. Existem aqueles meramente informativos, que não trazem nenhum compromisso com escalas ou famílias de projeções. Estes são os mapas apresentados nas páginas Internet que informam localizações de ruas, por exemplo. Mas existem também aqueles que primam pela precisão cartográfica. Neste grupo

se enquadram principalmente os trabalhos científicos ou técnicos, desenvolvidos com finalidades variadas mas que necessitam ter noção espacial acurada da localização de dados levantados em campo.

Os sistemas de informações geográficas (SIG) são os instrumentos usados por excelência para manipular informações espaciais, geralmente na forma de mapas. Estes sistemas foram criados há pelo menos trinta anos, adaptando a tecnologia de computadores aos já conhecidos milenares processos cartográficos. O computador foi o responsável para que técnicas manuais antigas e morosas da cartografia fossem agilizadas, além de permitir o armazenamento de mapas no formato digital, dispensando assim os caros ambientes climatizados para sua manutenção. Algo muito semelhante ao que acontece com livros e bibliotecas. Alem de permitirem os formatos digitais, os computadores possibilitaram a implementação dos métodos mecânicos de cruzamento destes mapas, entre outras funções, com o uso de programas. Hoje, estes sistemas são altamente sofisticados, com uma grande quantidade de funções de análise espacial, precisão cartográfica acurada, transformações entre diversas famílias de projeções, etc. Outra grande vantagem que estes sistemas incorporaram foi sua ligação com os sistemas de sensoriamento remoto, permitindo a junção de mapas digitais com imagens de satélites, abrindo novos campos em diversas áreas de pesquisa relacionadas às ciências da terra.

Estes sistemas são executados em computadores pessoais ou, no máximo, em licenças para funcionar em redes locais. O compartilhamento dos mapas, nestes casos, fica restrito apenas a estes ambientes. Uma funcionalidade básica em qualquer SIG é a interatividade usuário-sistema. Não se imagina um SIG em que o usuário não possa apontar o cursor para uma região do mapa e solicitar algum tipo de serviço (ampliar, reduzir, mostrar informações, etc.). Isto pode ser feito prontamente porque o SIG está instalado no computador do usuário. Mesmo em ambiente de rede local, esta tarefa é quase instantânea. Outra característica destes SIG era o variado formato de armazenamento de dados. Cada fabricante adotava, e ainda adota, um formato proprietário. No entanto, dois formatos estão se tornando comum para troca de informações entre diferentes SIG: SHAPEFILE da *Environmental Systems Research Institute* (ESRI) e DXF, da AutoCad.

Quando estes SIG eram desenvolvidos para equipamentos e sistemas operacionais específicos a aplicação procurava tirar o máximo de proveito dos seus recursos. Mesmo os compiladores eram projetados para permitir esta otimização. Tal condição não existe na Internet. Não se pode desenvolver

uma aplicação nem para um sistema operacional nem para uma máquina específica. Os arquitetos de sistemas se depararam com um novo paradigma, da interoperabilidade. Produtos para a Internet tinham que ser executados em qualquer tipo de máquina e sistema operacional. A interatividade, operação básica para quem usa um SIG, não dispõe da mesma facilidade na Internet. Enquanto no sistema tradicional a interatividade usuário-máquina é instantânea, tal não acontece na Internet, onde se trabalha com o conceito de cliente/servidor. Este conceito quebra as barreiras da proximidade física, podendo estar cada um em locais tão dispersos como diferentes continentes.

Com a expansão dos meios de comunicação no mundo inteiro, e com a Internet, muitos pesquisadores e técnicos encontraram uma nova oportunidade de publicar mapas resultantes de seus trabalhos de pesquisa. A meta linguagem *Hyper Text Markup Language* (HTML), com a opção <IMG SRC USEMAP> permitiu que mapas no formato matricial (GIF, JPEG) pudessem ser divulgados. Aliado a esta facilidade, o comando <AREA SHAPE> permitia que regiões destes mapas pudessem ser selecionadas e informações adicionais podiam ser anexadas, com o uso de endereçamento eletrônico (*hyperlinks*). Embora fosse um começo, tal opção de uso do HTML ainda era muito limitada, não permitindo a implementação de funções interativas comuns em um SIG. Além disto, deve-se considerar que apenas mapas matriciais (imagens) podiam ser transferidos pela rede. Mapas vetoriais não são manipulados pelo HTML. Recentemente, o consórcio *World Wide Web Consortium* (W3C) definiu um padrão de divulgação de mapas vetoriais na Web usando a especificação *Scalable Vector Graphics* (SVG), originalmente especificado pela Adobe como *Precision Graphics Markup Language* (PGML). Existe também o padrão da Microsoft, *Vector Graphics Markup Language*, mais conhecido pela sigla VML. Mas a opção para quem desenvolve aplicações para a Internet é o SVG, por não ser proprietário.

Empresas comerciais da área de SIG e grupos de pesquisa começaram a desenvolver aplicativos para permitir a publicação de mapas pela Web. Deste então, só tem crescido a oferta de sistemas que podem realizar esta tarefa (Miranda, 2002a). No meio de tantas opções, aparece a questão: o que usar? Primeiramente, vem a divisão entre sistemas comerciais e de domínio público. As opções comerciais, de fato, apresentam mais vantagens que as contrapartes livres. No entanto, para tarefas básicas, a opção por um programa livre não apresenta problemas e tem sido usado cada vez mais por empresas de todos os portes. Indiferente do sistema ser comercial ou de domínio público, existe uma coisa básica para ambos: em que plataforma

de programação ele foi desenvolvido. Os SIGs tradicionais foram desenvolvidos em linguagens de programação compiladas, como C, C++ ou Pascal, tanto para Windows como para Unix e suas variantes. Para a Internet, as opções são Java e aplicações usando CGI, conforme descritos a seguir.

Para abreviar e uniformizar algumas notações neste trabalho, adotou-se a seguinte convenção: *navegador* ou *cliente* significa um navegador Web, como Netscape, Internet Explorer ou Mozilla; *servidor* significa um servidor HTTP ou servidor Web, podendo ser o Internet Information Server da Microsoft ou Apache, da Apache Software Foundation, entre outros tantos disponíveis. *Sun* significa a empresa Sun Microsystems, detentora do Java. Em algumas situações, *cliente* pode também ser uma aplicação, como uma página HTML, sendo executada dentro do navegador, trazendo a situação de um *cliente* (página HTML) dentro de outro *cliente* (navegador). Caso semelhante acontece com *servidor*. Existe a situação do servidor HTTP estar executando outro servidor, no escopo desta publicação, um *servidor de mapas*. Para evitar qualquer confusão, a palavra servidor aqui sempre se refere a um servidor HTTP e cliente sempre se refere a um navegador. Casos específicos serão explicitamente citados.

Requisito Básico: Ferramentas Gráficas

Um requisito básico para se desenvolver um sistema para manipulação de mapas, entidades essencialmente vetoriais, são as primitivas gráficas que a linguagem de programação oferece.

Java, como era de se esperar, tem evoluído ao longo do tempo desde sua concepção. As primeiras primitivas gráficas vieram no *Abstract Windowing Toolkit* (AWT), como o desenho de formas geométricas básicas (retângulo, círculos, ...), tratamento de cor, etc. Mais recentemente, a *Sun* liberou um novo pacote gráfico, o Java2D API (*Application Programming Interface*), oferecendo novos recursos com melhorias em gráficos 2D, textos e algumas funções para processamento de imagens matriciais. Para a área de processamento de imagens digitais, existe um projeto mais avançado, o *Java Advanced Imaging* (JAI API), recomendado para áreas como:

- Defesa militar e serviços de inteligência.
- Processamento digital de imagens de satélite.
- Bioinformática.

- Pesquisa aplicada.
- Fotografia digital.
- Comércio eletrônico e ensino.

Conforme especificação da *Sun*, tal extensão se aplica bem no caso de tratamento de imagens digitais geradas por sensores, estejam eles a bordo de um satélite ao redor da terra ou em um aparelho hospitalar para tomografia. Dentre as funções de processamento de imagens encontram-se funções de convolução, cubo de cores, transformações geométricas, histograma, espaço de cores IHS (Intensity Hue Saturation) entre outros, interpolações bicúbica, bilinear, operação com vizinho mais próximo, operação pontual para transformação de valores de pixels, algoritmos de morfologia matemática, entre outras opções. Para quem tem interesse em desenvolver aplicativos nesta área, o JAI API é a recomendação apropriada. Porém, para SIG, a direção é outra. Os requisitos de tratamento de mapas em um SIG são bem diferentes do processamento digital de imagens, embora algumas funções desta área possam ser usadas em um SIG.

Desde o começo de seu desenvolvimento, o SIG foi projetado para trabalhar com um entre dois modelos básicos de dados: matricial ou vetorial. Cada um apresenta vantagens e desvantagens. Como eles são complementares, o que se tem hoje são sistemas capazes de fazer a migração entre estes modelos de dados. Contudo, no processo de migração matriz-vetor, ocorrem "avarias" ou perda de informação nos polígonos. Quando se trata de manipular dados pela Internet, o tempo de transmissão de dados entre cliente e servidor é um ponto crítico. Portanto, quanto mais compactos forem os dados, menor o tempo de transmissão. Dados matriciais ocupam mais espaço que dados vetoriais, mesmo considerando algoritmos de compactação extremamente eficientes existentes hoje.

Para sistemas geográficos pela Internet, não necessariamente um SIG, fica o desafio de se poder trabalhar com dados vetoriais. Este desafio ainda esta sendo resolvido, principalmente com a entrada do padrão *Scalable Vector Graphics* (SVG). Desenhar um gráfico em uma máquina cliente não é um processo tão simples como desenhar em um computador local. Além disto, o navegador Web para mostrar o gráfico no formato SVG tem que ter um *plug-in* da Adobe, que pode ser instalado gratuitamente. Enquanto a imagem não é carregada na sua totalidade, o aplicativo sendo executado no servidor está incapacitado de realizar qualquer manipulação nela, por exemplo, saber a sua dimensão.

Desenhando com Java

Quando se desenvolve aplicação em Java para trabalhar com imagens, existe o conceito do "observador da imagem", porque as imagens são processadas assincronamente, diferente da operação local, que é síncrona. Com assincronia, como saber se a imagem requerida já foi carregada? E se existiu um erro na carga da imagem? Estes, e outros, problemas são de responsabilidade do observador de imagens, objeto que implementa a interface `ImageObeserver`. Trata-se de um parâmetro que monitora o status da imagem e torna a informação disponível para o resto da aplicação. Quando uma imagem é carregada, o observador de imagem é notificado do progresso da operação, incluindo quando novos pixels estão disponíveis, quando um quadro completo da imagem está pronto e se existiu um erro durante a operação. O observador de imagem também recebe informações do atributo da imagem, como sua dimensão e propriedades (Niemeyer & Peck, 1997).

Suponha que se queira desenhar um mapa de solos com cinco polígonos. Em um SIG local, esta operação é simples. Basta escolher a opção de desenhar no menu, escolher o arquivo de dados e prontamente o SIG responde desenhando o mapa. Como seria esta operação na Web, usando a arquitetura cliente/servidor? Primeiro, a solicitação seria a mesma, escolhendo a opção desenhar na interface. Neste momento, acontece um ciclo de requisição-resposta entre cliente e servidor. Além da escolha de desenhar, o cliente enviaria na requisição o nome do arquivo para o servidor. O servidor então iria procurar pelo arquivo solicitado e pelo aplicativo que realizaria a operação de envio de resposta, que poderia ser um *servlet* ou um aplicativo CGI. Este aplicativo seria responsável por processar a ordem de desenho, pegando o arquivo que contém o mapa de solos, e efetivamente desenhando o mapa. Apresenta-se um exemplo de código em Java. Este código não seria diferente em um SIG local, feito em C++, por exemplo.

```
...
// Desenha arcos.  Os dados seriam lidos do arquivo solos.
// g é o contexto gráfico.

public class MapaPoligonal {
  int numArcos;                    // quantidade de arcos
  int [ ] corArco;                 // arranjo para guardar a cor de cada arco,
se necessário
  int [ ] numparxyArco;            // quantidade de pares (x,y) no arco
```

```
    int [ ][ ] arcoX, [ ][ ] arcoY;      // pares (x,y) do arco
    // Define ambiente gráfico
    Graphics g = null;
    Frame frame = null;
    …

    MapaPoligonal(file input) {
      // construtor da classe mapa poligonal
      // define número de arcos e suas coordenadas, cor do arco, entre outras coisas.
       ...
    }
     ...
    // método para desenhar arcos
    Image desenhaArcos() {

  frame = new Frame();
  frame.addNotify();
   Image imagem = frame.createImage(WIDTH, HEIGHT);
  g = imagem.getGraphics();
  // pega quantidade de pares (x,y) do primeiro arco
  int tamanhoArco = numparxyArco[0];
  // percorre todos os arcos
  for (int k = 0; k < numArcos; k++) {
      // define tamanho dos arranjos
      int coordenadasX [tamanhoArco];
      int coordenadasY [tamanhoArco];
     // percorre todos os pares (x,y) do arco e armazena seus valores
     for (int j = 0; j < tamanhoArco; j++) {
        coordenadasX[j] = arcoX[k][j];
        coordenadasY[j] = arcoY[k][j];
     }
     // define cor e desenha arco
     g.setColor(Color.corArco[k]);
     g.drawPolyline(coordenadasX, coordenadasY, —j);
     // pega tamanho do próximo arco
     tamanhoArco = numparxyArco[k];
  }
  return (imagem);
}
```

```
  ...
}

// Programa principal
file solos;

MapaPoligonal mapaSolos = new MapaPoligonal(solos);
...
// Gera imagem no formato GIF para mostrar no navegador
GifEncoder encoder = new GifEncoder(mapaSolos.desenhaArcos(), outputStream);
encoder.encode();
…
```

Depois de desenhar os arcos, o aplicativo, via servidor, tem que mandar a resposta para o cliente. A resposta tem que identificar o tipo de conteúdo para o cliente. Como não existe um tipo de conteúdo vetorial no HTML, o aplicativo tem que transformar o mapa vetorial em um matricial, usando um método de transformação para GIF, JPEG ou outro formato gráfico. No exemplo acima, a resposta é do tipo GIF. A resposta é enviada pela rede e mostra o mapa no monitor do cliente. Notar a grande diferença entre as duas aplicações para o desenho de um simples mapa. No SIG local, a consulta e processamento é feita no disco rígido local, e na web, a consulta e processamento é feita remotamente. Dependendo da velocidade da linha, este tempo pode variar, sempre para mais em relação ao SIG local.

Identificando as Aplicações

Interação Servidor/HTTP

Como *servlets* e aplicativos CGI são executados no lado do servidor, descreve-se brevemente o processo de comunicação deste com o cliente. No caso de aplicativos que são executados no lado do servidor, a tramitação servidor/cliente acontece da seguinte forma: o cliente faz uma requisição, o servidor responde e a transação está feita. Este é o padrão HTTP, que usa um tipo de protocolo chamado “sem estado” (Hunter & Crawford, 1998). Quando o cliente manda uma requisição, a primeira coisa que ele especifica é um comando HTTP chamado método, que diz ao aplicativo o tipo de ação que ele deve fazer. Os métodos mais usados são GET ou POST. Nesta requisição

também vai o endereço de um documento a ser usado e a versão do protocolo HTTP. Outras informações também podem ser enviadas pelo cliente, após este cabeçalho inicial. Depois que o cliente envia sua requisição, o aplicativo a processa e manda de volta uma resposta. A primeira linha desta reposta contém (i) o status que identifica a versão do protocolo HTTP que o servidor está usando, (ii) um código de status e (iii) uma descrição deste código, por exemplo:

HTTP/1.0 404 Not Found.

Neste caso, o servidor está usando a versão 1.0 do protocolo e dizendo que não encontrou a solicitação (documento) requisitado pelo cliente, retornando como código de status o valor 404. Após a linha de status, o servidor envia cabeçalhos de respostas que informam ao cliente parâmetros como o tipo de programa o servidor está executando e o tipo de conteúdo da resposta do servidor, por exemplo:

Date: Tuesday, 23-July-2003 11:56:12 GMT

Server: Apache/1.3.14 (Win32)

MIME-version: 1.0

Content-type: text/html

etc.

O cabeçalho do servidor fornece informações sobre o programa servidor, enquanto o cabeçalho `Content-type` especifica o tipo *Multipart Internet Mail Extension* (MIME) dos dados incluídos com a resposta. O que o cliente recebe depois, neste caso, são comandos HTML, que ele interpreta e mostra no navegador. Então, em última instância, um *servlet* ou um aplicativo CGI gera código HTML dinamicamente a pedido do cliente usando um formulário de requisição no navegador, como o comando < FORM >. Quando o servidor está enviando sua resposta, um campo importante é o que define o tipo de conteúdo (MIME), que pode ser um simples texto (text/html) ou um conteúdo do tipo multimídia, como uma imagem GIF (image/gif) ou JPEG (image/jpeg.)

De uma maneira ou de outra, o que um servidor manda se enquadra em um texto ou uma imagem, além de sons ou vídeos. Não há opção para mandar um "arquivo vetorial." E esta é uma limitação de se desenhar mapas vetoriais com servidores. O mesmo não acontece com *applets*. Estes programas podem desenhar mapas vetoriais, pois são aplicações do tipo cliente. Trabalhando na internet, deve-se estar ciente de que os dados devem trafegar pela rede. Enquanto uma aplicação local dispõe dos dados no disco rígido do computador local, ou, no máximo, em outro disco de uma rede local, na Internet o arquivo de dados pode estar em outro país. Dependendo da velocidade de comunicação da linha, pode levar alguns minutos para sua carga, o que representa uma infinidade de tempo nos dias atuais.

Aplicativo CGI

As primeiras páginas construídas para a Internet eram escritas em HTML, criando conteúdo estático. A geração de páginas com conteúdo dinâmico só foi possível com o surgimento de novas técnicas, como o uso da interface comum de comunicação — *Common Gateway Interface* (CGI). Em uma aplicação Web tipo CGI, um usuário faz uma solicitação ao navegador, que a repassa ao servidor. O servidor identifica a requisição como pertencendo a um CGI e a repassa novamente a um programa externo — o aplicativo CGI — que de alguma maneira implementa uma funcionalidade de resposta, que é então enviada de volta ao cliente, via servidor. O advento do CGI permitiu a implementação de uma grande variedade de funcionalidades nas páginas Web (Hunter & Crawford, 1998).

Todos os servidores permitem implementação de CGI. E o que significa CGI? Um padrão para comunicar aplicações externas (cliente) com um servidor de informação, tal como HTTP ou servidor Web. Uma aplicação CGI pode ser escrita em qualquer linguagem de programação, compilada ou interpretada. As linguagens compiladas freqüentemente usadas são C ou C++, dado que mais pessoas estão sendo treinadas no uso destas linguagens. E as interpretadas são PERL e PHP. Outras também podem ser usadas, de acordo com a experiência do programador. O ciclo de vida de um CGI requer que o servidor crie um novo processo para executar cada requisição, passando toda informação para o CGI via variáveis de ambiente e entrada padrão. Esta é uma desvantagem, pois criar um processo para cada requisição consome muito tempo e recursos do servidor. Alem disto, um programa CGI não pode interagir com o servidor e assim tirar proveito de

suas funcionalidades, pois ele é executado em processos separados, fora do servidor.

A tarefa de uma aplicação Web escrita em CGI é permitir que o usuário possa ter uma página Web dinâmica, possibilitando uma maior interatividade entre ele e o conteúdo da página. De qualquer maneira, a resposta que um CGI envia ao cliente tem que estar no padrão que ele entende, o HTML. Isto significa dizer que um servidor de mapas escrito em CGI não pode enviar mapas vetoriais como resposta, mas apenas os formatos permitidos no tipo de conteúdo MIME. Para desenvolver um aplicativo CGI, são necessários dois recursos: um servidor HTTP e uma linguagem de programação, que pode ser compilada, como C/C++, ou interpretada, como PERL ou PHP. O servidor mais usado é o Apache[1]. Durante a instalação do Apache, vários diretórios são criados. Entre estes, encontra-se o "cgi-bin." É neste diretório que todas as aplicações CGI são colocadas.

O segundo recurso é o código executável de um programa no diretório cgi-bin. Para testar o código, primeiro inicia-se o Apache, abre-se um navegador e na linha de endereço fornece-se uma chamada ao programa. Supondo que se desenvolveu um programa em C, mathcalc.c, sua execução pela rede seria iniciada com a chamada:

http://localhost/cgi-bin/mathcalc.exe

Servlet

Servlet é uma aplicação Java, uma extensão genérica do servidor, implementada como uma classe Java que pode ser carregada dinamicamente para expandir a funcionalidade do servidor (Hunter & Crawford, 1998). Por ser executado no servidor, ele apresenta características próprias. Dentro do servidor, ele é executado em uma máquina virtual Java (*Java Virtual Machine* - JVM), sendo seguro e portável. Como ele é executado dentro do domínio do servidor, não requer suporte Java no cliente. Aplicações com *servlet* geralmente são projetadas e implementadas para uso com um mecanismo próprio de *servlet*. Pelo fato de ele ser executado no servidor, pode realizar tarefas que não são possíveis a um aplicativo CGI.

---

[1] Para obter uma cópia deste programa, acessar o endereço http://apache.org, escolhendo a opção HTTP Server.

Um *servlet* apresenta algumas vantagens, como portabilidade. Como ele é escrito em Java, pode ser executado em qualquer sistema operacional. Por conta da portabilidade, o *servlet* evita o uso do *Abstract Windowing Toolkit* (AWT). Com isto, interfaces gráficas não podem ser implementadas por um *servlet*. Mas pode-se requisitar um serviço de servlet a partir de uma interface gráfica. Outra vantagem, a chamada a um *servlet* é eficiente. Uma vez que ele é carregado, permanece na memória do servidor, sendo acessado por uma chamada simples e requisições concorrentes e múltiplas são manuseadas por cópias (*threads*) separadas. Outro ponto a favor é a segurança. Ele pode manusear erros de maneira segura, através do mecanismo de administração de exceções do Java. O próprio servidor está protegido dos *servlets* pelo uso do gerenciador de segurança Java. Um *servlet* não pode, por exemplo, danificar o sistema de arquivos do servidor (Hunter & Crawford, 1998).

Para desenvolver um *servlet* são necessários dois requisitos: os arquivos com as classes Servlet API e um mecanismo de *servlet*. As classes podem ser adquiridas na página www.javasoft.com, escolhendo a opção J2SE (Java 2 Standard Edition). Um mecanismo de *servlet* bastante usado é o Tomcat, que pode ser adquirido gratuitamente na página do Apache Jakarta Project http://jakarta.apache.org, sob as opções "Products - Tomcat." Tendo escrito o *servlet*, sua execução é simples. Basta executar o Tomcat, abrir um navegador e realizar a chamada:

http://localhost:8080/exemplo/meuServlet

Os *servlets* geralmente são colocados sob o diretório *webapps* criado durante a instalação do Tomcat. No caso acima, um subdiretório *exemplo* foi criado embaixo do *webapps* e o *servlet* foi denominado *meuServlet*.

Applet

Finalmente, o terceiro tipo de aplicação Web com o qual se pode desenvolver programas para manipulação de mapas é um *applet. Applet* é também uma aplicação escrita na linguagem de programação Java. Na verdade, *applets* são pequenos programas escritos em Java e incluídos em páginas Web. Este termo foi criado pela equipe da *Sun* que desenvolvia o Java. Para um navegador executar um *applet*, ele deve ter embutido uma máquina virtual

Java (JVM), o que acontece com todos os navegadores disponíveis. Uma máquina virtual Java nada mais é que um interpretador de código de execução Java, conhecido pela extensão .CLASS.

Os *applets* podem ser executados no computador local ou a partir de um computador remoto, geralmente disponibilizado através de um servidor. Se o *applet* a ser executado está no próprio computador, o navegador nem precisa estar conectado à Internet para o executar. Porém, se o *applet* não estiver disponível no computador local, será necessário uma conexão Internet para que ele seja carregado a partir de um *Uniform Resource Locator* (URL) especificado (Walnum, 1997). *Applets* são eficientes para desenhar gráficos e implementar interfaces gráficas. Desenhos gráficos são feitos com o uso de classes gráficas definidas no AWT API ou no Java 2D API. Interfaces gráficas podem ser feitas com o AWT ou o SWING, uma extensão mais nova do Java que implementa facilidades para desenho de interfaces gráficas.

Para um *applet* ser executado as exigências são menores que um CGI ou *servlet*, ele não precisa de um servidor. A única exigência é o compilador Java e o navegador. Escreve-se um programa Java, fazendo dele uma extensão (subclasse) da classe *Applet*. Depois de compilado, sua execução pode ser iniciada no navegador através de uma chamada de dentro de um código HTML, como:

```
< HTML>
        < HEAD>
                < TITLE > Executando um Applet< /TITLE>
        < /HEAD>
        < BODY>
                Exemplo de Applet.
                < P>
                < APPLET code=”MathCalc.class” width=450 height=150>
                < /APPLET>
        < /BODY>
< /HTML>
```

Pode-se também executar um *applet* usando o programa *appletviewer*, disponível no pacote J2SDK do Java 2 Standard Edition.

Para um applet ser executado, todos os dados necessários precisam ser carregados do seu local de origem. Isto significa que se um *applet* precisar de muitas imagens e dados para ser executado, ele só pode ser executado no cliente após todos eles serem carregados. Se a aplicação pretende disponibilizar um grande número de mapas, então deve-se estar ciente de que a carga dos dados pode demorar. Se o usuário estiver usando uma linha discada para executar o *applet*, este tempo pode ser grande, inviabilizando o uso da aplicação.

Servidores de Mapa de Domínio Público

Neste item analisam-se três aplicativos que podem ser adquiridos gratuitamente na Internet para publicar mapas pela Web, cada um pertencendo a uma das categorias analisadas anteriormente. O uso de um deles vai depender da quantidade de mapas e dados que se pretende publicar pela rede.

Como as aplicações *servlet* e CGI aqui apresentadas implementam a especificação *Web Map Server* (WMS) do OpenGIS Consortium (OGC), o próximo tópico é dedicado a descrever seus pontos básicos, de acordo com o *CookBook* da OCG (Kolodziej, 2003).

Especificação WMS

A especificação WMS padroniza a maneira na qual clientes requisitam mapas. Clientes requisitam mapas de uma instância WMS em termos de planos de informação e providenciam parâmetros tais como tamanho do mapa a retornar e sistema de referência a ser usado para o desenho do mapa (Buehler, 2003). O mérito da iniciativa OpenGIS está em elaborar uma abordagem de mapas pela Web usando um padrão de dados não-proprietário, baseado em interfaces abertas, codificações e esquemas. A palavra chave da iniciativa é interoperabilidade de serviços de geoprocessamento, dados e aplicações. Interoperabilidade significa intercomunicação no nível de protocolo de comunicação, equipamentos, programas e compatibilidade no plano dos dados (Kolodziej, 2003). As especificações WMS são baseadas na linguagem de descrição de dados

EXTENDED MARKUP LANGUAGE (XML) do W3C. Ressalta-se que empresas comerciais de grande porte também estão seguindo a especificação WMS. A Intergraph a usa no GeoMedia e a ESRI no ArcIMS, aplicativos que disponibilizam mapas na Web.

Estas especificações tratam de serviços básicos Web, acesso a imagens, visualização, manipulação e capacidade de transformação de coordenadas. Elas especificam os protocolos de requisição e resposta para interações abertas cliente/servidor de mapas baseadas na Web. No contexto do WMS, um mapa é uma imagem matricial. Conforme visto, um *servlet* ou um CGI não enviam mapas no formato vetorial, pelo menos por enquanto, mas apenas uma página HTML, com imagem matricial, nos formatos GIF, JPEG, PNG, etc. O que acontece, por exemplo, quando o cliente faz uma requisição para sobrepor mapas? O *servlet*/CGI pega a requisição, efetua a operação, codifica o resultado em uma imagem matricial e envia a resposta para o cliente. Este procedimento foi exemplificado no código apresentado no item "Desenhando com Java." Espera-se que esforços sejam direcionados para uma integração com SVG, e a resposta poderia ser enviada como um mapa vetorial. Isto pode ser feito, mas com uma condição: o navegador tem que ter instalado o *plug-in* para mostrar arquivos SVG.

Uma versão de servidor de mapas para Web em SVG, está disponível na página http://www.carto.net/projects/open_svg_mapserver, escolhendo a opção OpenSVGMapServer 1.01. Este aplicativo ainda está na forma de protótipo. É um programa grátis que permite distribuir mapas na Internet no formato SHAPEFILE do ArcView. Para usá-lo, é necessário instalar o MySQL, PHP e o ArcView. Uma outra limitação, o programa só pode ser executado no Internet Explorer 4.0+, contendo o *plug-in* da Adobe para visualizar os mapas gerados no formato SVG.

A especificação WMS para a arquitetura cliente/servidor, apresentada na Fig. 1, conta com três componentes: o cliente WMS, o servidor de mapas WMS e a base de dados. O cliente WMS é um programa que processa as requisições do usuário e visualiza dados espaciais. É uma série de páginas HTML geradas dinamicamente e executadas dentro do navegador, as quais se comunicam diretamente com um servidor de mapas via um protocolo HTTP através de três interfaces de requisição WMS, `GetCapabilities`, `GetMap`, e `GetFeatureInfo`. Pela proposta WMS, podem existir dois tipos de implementação de cliente: simples ("thin"), constando apenas de uma página HTML que usa o navegador para manipular interações com o usuário. As requisições WMS do cliente para o servidor são feitas com intermediação

do navegador na forma de URLs HTTP padrão. O cliente Web recebe a resposta do servidor, via navegador, na forma de mapas imagem em formato GIF, JPEG, PNG, XML ou outro formato codificado MIME.

Portanto, na forma cliente simples, as interações são realizadas no formato padrão de interação cliente/servidor com a repetição da seqüência requisição-resposta. Neste formato, a interação do usuário com o mapa pode ser prejudicada, pois depende da velocidade de comunicação do cliente com o servidor. As interações sob demanda são conseguidas pela construção de páginas HTML através da implementação de um CGI, um *Active Server Page* (ASP), um *Java Server Page* (JSP) ou um *servlet*, desempenhando o papel de servidor de mapas WMS.

O outro tipo de cliente é o "inteligente" ("*thick*"), que implementa no navegador aplicações Java, como *applets* interagindo com o servidor de mapas, *Javascript* ou *plug-ins*. Este tipo de cliente implementa facilidades muito além dos simples URLs. A vantagem do cliente simples em relação a este é seu baixo custo operacional e facilidade de poder ser visto em qualquer navegador. A única desvantagem é que a interação com o usuário fica bastante restringida. O uso de uma ou outra opção vai depender da demanda do projeto a ser usado.

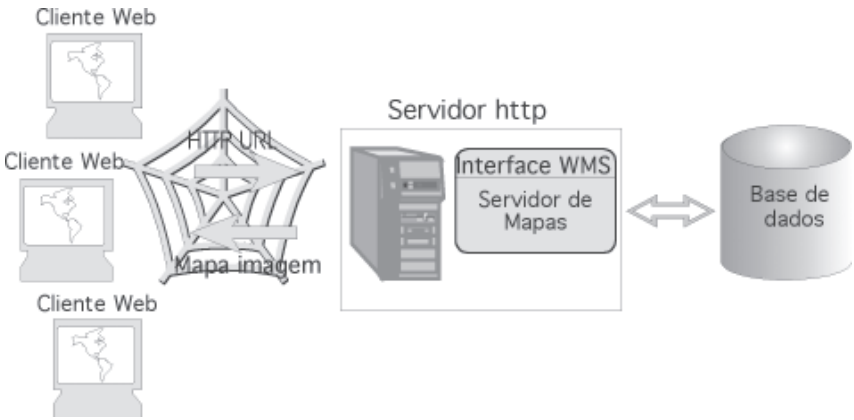


Fig. 1. Configuração básica da especificação WMS cliente/servidor.

Como mostra a Fig. 1, uma interação típica cliente/servidor WMS começa com o cliente WMS requisitando um `GetCapabilities` do servidor de mapas WMS de maneira a determinar o que ele pode fazer e quais mapas ele pode oferecer. Em seguida, o cliente WMS requisita um mapa imagem de acordo com estas capacidades, usando o `GetMap`. Finalmente, o cliente WMS pode escolher um ponto no mapa imagem e requisitar mais atributos acerca deste ponto, através do `GetFeatureInfo`.

Em resposta a uma requisição `GetCapabilities`, o servidor WMS produz um documento XML contendo os serviços disponíveis - capacidades - como um metadado, descrevendo todas as operações permitidas e informações sobre os mapas disponíveis. A aplicação cliente WMS tem que analisar o documento XML para recuperar as informações necessárias para requisição dos mapas. Com estas informações, o cliente WMS pode requisitar um mapa imagem do servidor WMS usando a operação `GetMap`. Em seguida, o servidor WMS processa a requisição, normalmente acessando dados armazenados em uma base de dados. Por fim, o servidor WMS retorna ao cliente WMS um mapa imagem codificado na forma de uma figura MIME, como GIF ou JPEG. Com o mapa no navegador, o cliente WMS pode requisitar dados de atributos do mapa escolhendo um ponto no mapa, usando a função `GetFeatureInfo`. A resposta do servidor WMS pode vir nos seguintes formatos: arquivo *Geography Markup Language* (GML), um arquivo texto ou um arquivo HTML. O cliente WMS analisa o GML e mostra os atributos requisitados do mapa no navegador.

O servidor de mapas WMS, portanto, atende às solicitações do cliente WMS. Um servidor de mapas é um componente de programa que libera gráficos em uma interface comum para um cliente. Estes servidores aceitam requisições de clientes como URLs e retornam resultados codificados como GIF, JPEG, PNG, XML e outros. Geralmente, um servidor de mapas está ligado com um servidor Internet que controla as requisições do cliente via HTTP. O servidor de mapas tem como objetivo disponibilizar informações geográficas tipicamente dividida em níveis temáticos, informações estas contidas em um arquivo de metadados do tipo XML.

Por fim, a base de dados, cuja função é armazenar dados espaciais e não-espaciais, processando requisições do servidor de mapas. Esta base armazena atributos geográficos dos dados que podem ser acessados pela interface WMS, gerando mapas e/ou documentos GML. Os resultados de consultas da base de dados são primeiramente processadas pelo servidor de mapas e então transmitidos para a aplicação cliente. Atualmente, os servidores de mapas disponíveis trabalham com sistemas gerenciadores de banco de dados que implementam o padrão *Structured Query Language* (SQL), com o Oracle, Access, MySQL, PostgreSQL/PostGIS, etc. PostGIS é

um banco de dados que armazena dados vetoriais. Trata-se de uma extensão do banco de dados relacional orientado a objetos PostgreSQL, que permite que objetos SIG sejam armazenados no banco de dados. Ele suporta os atributos básicos definidos pelo OCG, `Point`, `LineString`, `Polygon`, `MultiPoint`, `MultiLineString`, `MultiPolygon` e `GeomCollection` (Kolodziej, 2003).

Servidor de Mapas Tipo Servlet: ALOV

O ALOV Map é uma iniciativa da organização sem fins lucrativos ALOV (www.alov.org) e existe em duas versões: cliente e cliente/servidor. Informações sobre a versão cliente, um *applet*, podem ser encontradas em Miranda (2002b). A versão *servlet* do ALOV implementa a especificação WMS, do OpenGIS Consortium. O ponto de partida para sua utilização é a definição do CLEARINGHOUSE, um serviço para armazenar as informações dos mapas em uma base de dados. Uma variedade de gerenciadores de banco de dados pode ser usada, como Access, MySQL, Interbase, Hypersonic e Oracle. Dependendo do gerenciador, será necessário definir o driver JDBC (*Java DataBase Connectivity*) específico para usá-lo com o *servlet*. Driver JDBC é um programa que implementa uma interface de comunicação entre os comandos SQL definidos dentro do *servlet* com os comandos SQL do gerenciador de banco de dados, seja ele qual for.

Ao construir a base de dados, com a operação identificada como *clearinghouse* pelo ALOV, o usuário carrega todos os mapas no servidor SQL. É necessário que esta base de dados cartográfica esteja em um dos dois formatos: SHAPEFILE ou MIF. O SHAPEFILE é o formato do ArcView. E o MIF é o formato do MapInfo. Tanto um como outro formato conta sempre com uma dupla de arquivos "shp + dbf" ou "mif + dbf". O primeiro arquivo sempre carrega as informações espaciais sobre o mapa e o segundo é um arquivo de atributos dos elementos geográficos contidos no primeiro. Este é o formato padrão da arquitetura de mapas vetoriais.

A Fig. 2 mostra uma tela do *servlet* "wms" do ALOV sendo executado, apresentando o mapa mundial. A versão cliente do WMS é uma implementação simples (*thin*), com apenas uma página HTML e a interatividade com o mapa é definida através de URLs. Esta página está definida no arquivo "wms_template.html." Por exemplo, apontando o cursor para a seta na direção leste, o URL é:

http://localhost:8080/Alovmap/wms?BBOX=-180.0,-45.0,177.30673,45.0&TOOL=1&REQUEST=GetPage&theme=m0&PROJECT=world.xml&FORMAT=image.gif&HEIGHT=400&WIDTH=400&LAYERS=Countries,Rivers,Cities

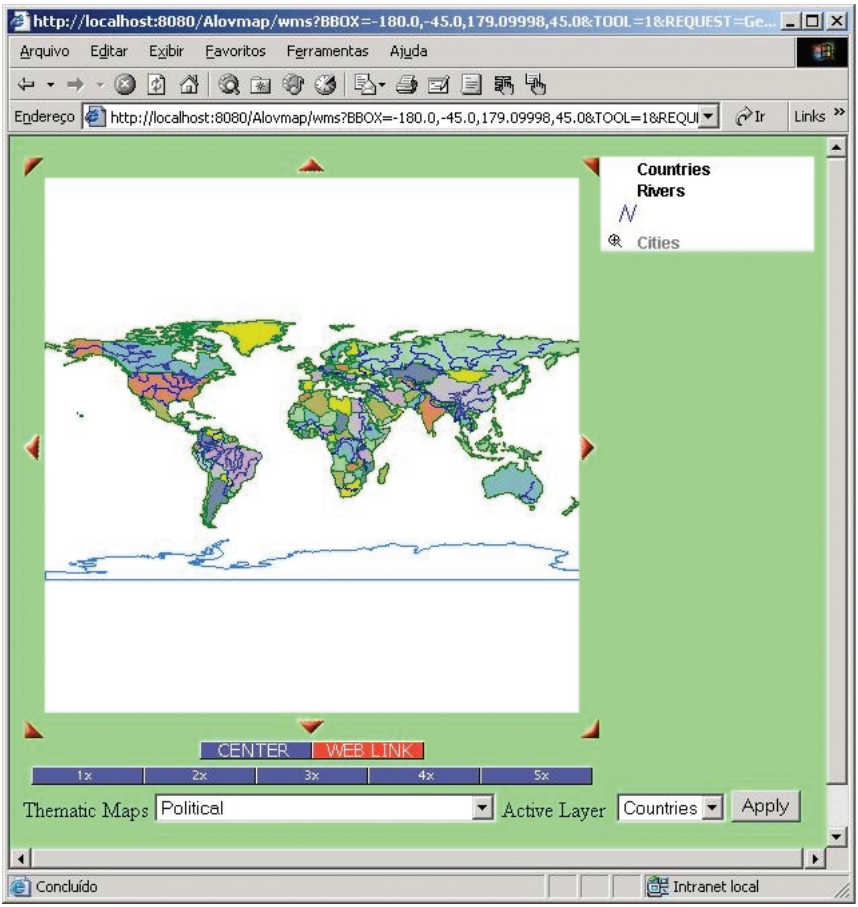


Fig. 2. Exemplo de saída do *servlet* ALOV.

Todos os dados após a interrogação são parâmetros passados para o *servlet*. Ao se escolher esta opção, o *servlet* vai gerar o mesmo mapa, porém, deslocado para a direita, dentro das coordenadas que definem um retângulo (BBOX) com o vértice superior esquerdo em (-180.0, -45.0) e o vértice inferior direito em (177.30673, 45.0). A requisição é um comando "GetPage", cujas especificações estão no arquivo "world.xml", definido pelo parâmetro PROJECT. A imagem é retornada no formato GIF, conforme especificado no parâmetro FORMAT, com altura e largura de 400 pixels, parâmetros HEIGHT e WIDHT, respectivamente. Três níveis, ou planos de informação, são gerados: países, rios e cidades, conforme parâmetro LAYERS. Os parâmetros não precisam aparecer nesta ordem. Outra observação, a linha de comando é contínua, só foi dividida por questão de espaço.

Alem da tela central que mostra o mapa, o ALOV permite que se defina a legenda do mapa e a barra de aumento, 1 a 5 vezes. O usuário pode definir mais de um mapa temático. No exemplo, aparece o mapa político. O usuário pode escolher um dentre os temas definidos, no caso, países, rios e cidades. Ao lado desta opção aparece o botão "*Apply*." Este botão é que permite a interatividade do usuário com o mapa. Quando o usuário muda sua opção de mapa temático e/ou plano ativo, deve selecionar este botão com o cursor. Neste momento, uma requisição é enviada ao *servlet*, que a executa e manda a resposta na forma de um mapa imagem.

Servidor de Mapas Tipo Aplicativo CGI: MapServer

O MapServer (http://mapserver.gis.umn.edu/) é um CGI desenvolvido em C++ pela Universidade de Minnesota de acordo com as especificações WMS. Portanto, uma implementação equivalente ao ALOV. A diferença entre os dois está na tecnologia utilizada para sua implementação, C++ e Java. O uso do programa é livre. Além disto, seus desenvolvedores usaram outras plataformas de código aberto ou livre para desenvolver o sistema, como Shapelib (http://gdal.velocet.ca/projects/shapelib), FreeType (http://www.freetype.org), Proj.4 (http://www.remotesensing.org/proj), libTIFF (http://www.libtiff.org), Perl (http://www.perl.com) entre outros.

O aplicativo pode ser executado nas plataformas Linux e Windows NT/9x sob um servidor http, por exemplo, Apache. Uma vantagem do MapServer, segundo seus idealizadores, é que ele suporta MapScript, permitindo que linguagens *scripts*, como Perl, PHP, Pyton acessem a API C do MapServer. Uma vantagem do MapScript é permitir ao usuário desenvolver aplicações, desde que os dados possuam uma componente espacial que possa ser acessada por uma linguagem *script*. Isto não significa necessariamente facilidades para o usuário, pois o desenvolvimento de *scripts* pode se tornar uma tarefa difícil.

O MapServer trabalha basicamente dando suporte ao formato vetorial SHAPEFILE do ArcView. A manipulação de formatos matriciais abrange um número maior de opções, como TIFF/GeoTIFF, GIF, PNG, ERDAS, JPEG e EPPL7. O formato de saída do aplicativo pode ser personalizado, a construção automática de legenda e barras de escala, a construção de mapas temáticos usando expressões lógicas ou regulares baseadas em classes, entre outras facilidades, podem ser efetuadas. Como está explicitamente escrito na página de acesso, o MapServer não é um SIG, no conceito pleno de todas as

facilidades que um SIG oferece. O que ele faz é prover condições suficientes de suporte para uma grande variedade de aplicações espaciais na Web.

O MapServer é uma aplicação do tipo servidor de mapas para disponibilizar mapas e imagens de satélite na Web, executada a partir de um servidor HTTP. Para ser executado, o MapServer necessita dos seguinte recursos:

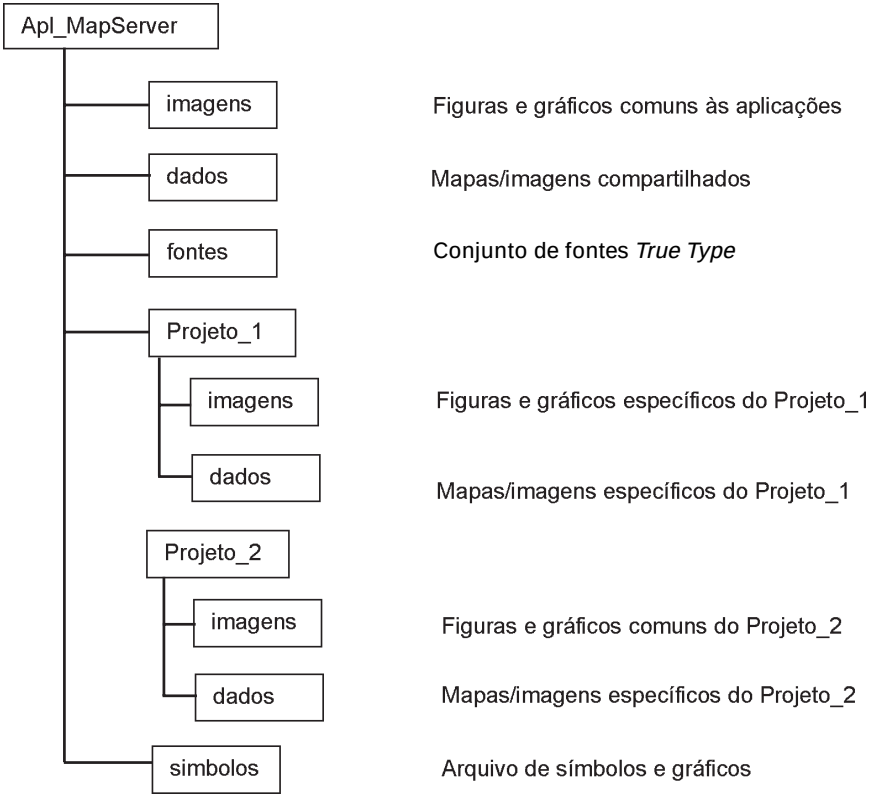
1. um servidor http;
2. um arquivo de apresentação do projeto, em HTML, com um texto descritivo sobre a aplicação com o MapServer (opcional);
3. um arquivo de configuração em formato texto, chamado de "*map file*", extensão .map, contendo uma descrição de todos os planos de informação (mapas temáticos) e seus parâmetros, como sistema de referência, nome do arquivo, rótulos a serem usados com o mapa, escala, cor, etc. As informações contidas neste arquivo serão usadas pelo MapServer quando estiver atendendo solicitações sobre os mapas;
4. o cliente WMS, uma implementação do tipo simples (*thin*) da especificação WMS, escrito em HTML;
5. o servidor WMS, implementado em C+ no programa MapServer; e
6. arquivos de dados, matriciais e/ou vetoriais, de um SIG.

Para usar o MapServer, a primeira tarefa é instalar uma versão estável, dado que existem versões beta, em fase de teste. Isto pode ser feito entrando na página http://mapserver.gis.umn.edu/. Algumas opções são mostradas nesta página, mas para usar o sistema, apenas duas são necessárias, *download* e *documentation*. As outras podem ser visitadas para informações adicionais. Para usuários do sistema operacional Windows existem duas alternativas, instalar a versão executável, mapserv.exe, ou instalar os fontes e compilá-los, gerando o executável (http://mapserver.gis.umn.edu/doc36/win32_compile-howto.html). Para usuários de alguma versão do Unix, é necessário baixar os fontes e compilá-los, para gerar o executável (http://mapserver.gis.umn.edu/doc36/unix-install-howto.html). Como toda aplicação do tipo CGI, o módulo executável do MapServer é instalado no diretório cgi-bin do servidor HTTP, sendo seus arquivos de dados armazenados no diretório de documentos do servidor HTTP, geralmente, um diretório denominado htdocs.

Para publicar mapas na Internet usando o MapServer, o primeiro passo é planejar como e onde colocar os arquivos requeridos pelo MapServer. Estes arquivos são definidos em três categorias:

- Arquivos específicos do MapServer: arquivo de configuração (.map) e de interface HMTL também conhecido como cliente WMS.
- Arquivos complementares: usados para dar forma estética a aplicação; não são usados pelo MapServer. Geralmente são gráficos ou imagens em arquivos HTML.
- Arquivos SIG: são os mapas vetoriais (SHAPEFILE) e imagens matriciais, geralmente imagens de satélite, no formato GeoTIFF.

Estes arquivos devem fazer parte de uma hierarquia de diretórios como mostrado no esquema:



Este tipo de hierarquia deve ser usado quando se tem vários projetos para disponibilizar mapas na Web. Para um só projeto, a estrutura pode ser simplificada, com o diretório principal identificando o projeto e quatro subdiretórios: imagens, dados, fontes e símbolos.

A Fig. 3 mostra a interface básica do MapServer. Ela não é muito diferente do *servlet* da ALOV, Fig. 2. Perceber no campo Endereço do Internet Explorer como a chamada do programa é diferente da anterior:

http://localhost/cgi-bin/mapserv.exe?
mode=browse&layer=lakespy2&layer=dlgstln2&zoomdir=0&zoomsize=2&imgxy=299.5+299.5&imgext=406051.291953+5227648.862813+518839.996573+5340410.720123&map=C:\Arquivos+de+programas\Apache+Group\Apache\htdocs\itasca\demo.map&savequery=true&program=\cgi-bin\mapserv.exe&map_web_imagepath=c:\Arquivos+de+programas\Apache+Group\Apache\htdocs\tmp\&map_web_imageurl=\tmp\&ref.x=58&ref.y=55

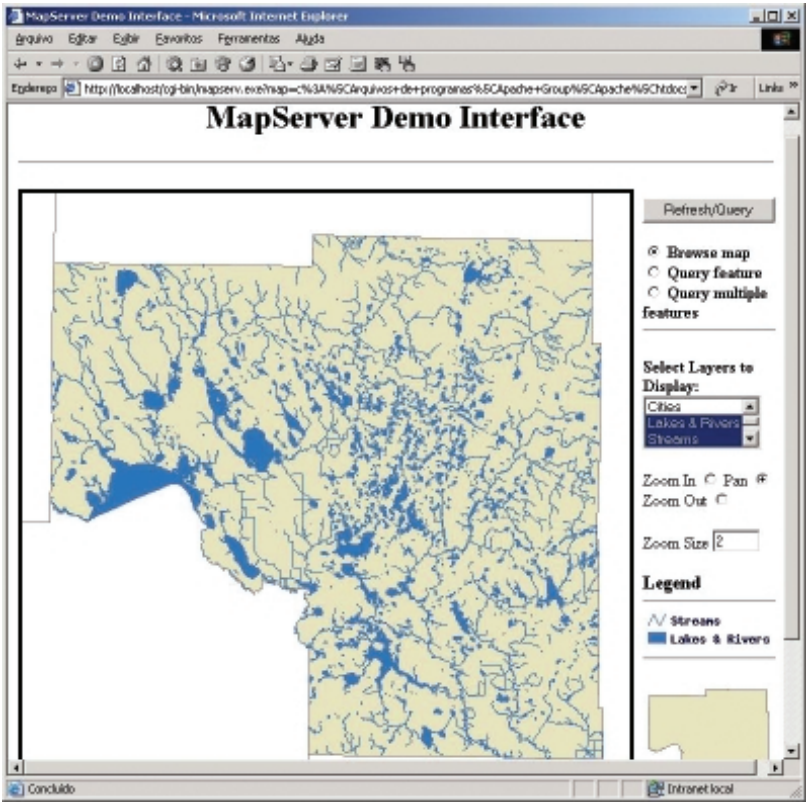


Fig. 3. Exemplo de saída do CGI MapServer.

O programa mapserv.exe é chamado a partir do diretório cgi-bin, neste caso, do Apache. Da mesma forma que aconteceu com o *servlet*, todos os dados após a interrogação são parâmetros passados para o CGI. Cada parâmetro é separado pelo símbolo "&". Por exemplo, o parâmetro "mode" indica o modo em que o sistema está operando, no caso, "browse", ou visualização. Duas outras opções podiam ser especificadas, conforme mostra a interface: "Query feature" ou "Query multiple features." O plano de informação visível é o "lakespy2", do parâmetro "layer." O arquivo de configuração (.map), necessário para iniciar o servidor de mapas, é definido pelo parâmetro "map", que fornece o caminho até ele, no caso, C:\Arquivos de programas\Apache Group\Apache\htdocs\itasca\demo.map.

Por exemplo, o texto a seguir mostra uma parte deste arquivo de configuração. O símbolo "#" significa comentário. A primeira especificação define parâmetros gerais do arquivo de configuração. Em seguida, as informações específicas para o plano de informação mostrado na Fig. 3, "lakespy2":

```
#
# Início do "map file"
#
NAME DEMO
STATUS ON
SIZE 600 600
EXTENT 388107.634400379 5203120.88405952 500896.339019834
5310243.30613897
UNITS METERS
SHAPEPATH "data"
IMAGECOLOR 255 255 255

        ...

#
# Definição do plano de informação (layer) "lakespy2"
#
LAYER
  NAME lakespy2
```

```
  TYPE POLYGON
  STATUS OFF
  DATA lakespy2

  CLASS
    NAME 'Lakes & Rivers'
    TEMPLATE "lakespy2.html"
    COLOR 49 117 185
  END

  HEADER "lakespy2_header.html"
  FOOTER "lakespy2_footer.html"

  TOLERANCE 3

  DUMP TRUE # permite exportar em GML

  METADATA
    WMS_TITLE "Lakes and Rivers"
     WMS_ABSTRACT "DLG lake and river polygons for Itasca County. See
http://deli.dnr.state.mn.us/metadata/full/dlglkpy2.html for more information."
    WMS_SRS "EPSG:26915"
  END # metadata
END #  LAYER lakespy2

          ...

END # map file
```

Este arquivo de configuração procura adequar a aplicação MapServer às normas de especificação do WMS, conforme visto anteriormente. Pelo exposto, tanto a aplicação tipo *servlet*, do ALOV, como a CGI do MapServer se adequam às especificações WMS, cada uma com suas peculiaridades, conforme mostra a Tabela 1.

Tabela 1. Exemplos de implementação da especificação WMS.

Implementação	*Metadado*	*Cliente WMS*	*Servidor WMS*	*Base de dados*
ALOV	Arquivo XML	Simples	Java Servlet	Padrão SQL
MapServer	Arquivo TXT	Simples	CGI C++	SHAPEFILE

Servidor de Mapa Tipo Applet: Berkeley GIS4.0

Por último, apresenta-se um aplicativo com facilidades para publicar mapas pela Internet do tipo cliente, o GIS-4.0 ou GIS Viewer. O GIS-4.0 é um *applet*, desenvolvido na Universidade da Califórnia em Berkeley. A Universidade libera apenas a versão executável do programa. O GIS-4.0 é uma ferramenta para Web que mostra e manipula conjuntos de informações geográficas, sejam eles planos de informação pontuais, vetoriais ou matriciais. Como foi dito, por ser uma aplicação do tipo cliente oferece facilidades de interação com os mapas e imagens, mas por ser um *applet*, todos os mapas e atributos devem primeiro ser carregados do servidor para serem utilizados. Um artifício para diminuir o tamanho dos dados é compactá-los em um arquivo do tipo "zip" ou "arj" e usar o parâmetro "archive" na chamada do *applet*. O GIS-4.0 pode ser adquirido gratuitamente no endereço http://elib.cs.berkeley.edu/gis.

Algumas das funcionalidades do GIS-4.0 incluem:

- consulta a banco de dados;
- visualização e manipulação de múltiplos planos de informação, permitindo habilitar/desabilitar diferentes planos;
- grande variedade de formatos de dados;
- mudanças de projeção baseadas em fatores de escala;
- pontos e regiões podem ser associados a diferentes URLs;
- conversão automática de projeções entre UTM, lat/long e Albers;
- operação de salvamento de arquivos e anotações nos mapas.

Operações de anotação permitem que o usuário inclua gráficos no plano de informação ativo. Para realizar esta tarefa, escolhe-se o menu "Layers" na barra de ferramentas. Os seguintes elementos gráficos podem ser desenhados: ponto, círculo, retângulo e linha vetorial. Com a opção linha

vetorial, o usuário pode digitalizar por cima de determinado mapa ou imagem que esteja ativo. O menu “Layers” permite também a inclusão de um ponto, círculo ou retângulo com um link. Estas opções estão pré-configuradas neste menu. Outras opções que este menu oferece incluem a remoção de plano de informação ou alteração de suas propriedades.

Como aconteceu com os aplicativos descritos anteriormente, um arquivo chave a ser fornecido pelo usuário é o de configuração. Nele, o usuário especifica informações adicionais para a aplicação saber o que e como fazer no manuseio dos mapas e imagens. Não é diferente com o GIS-4.0. Enquanto no ALOV ele era um arquivo XML e no MapServer um arquivo texto, no GIS-4.0 é um arquivo HTML. O texto a seguir mostra uma parte de um arquivo de configuração:

```
<html>
<header>
<title> GIS Viewer Exemplo 1 </title>
</header>

<body bgcolor=#FFFFFF>
<h2> GIS Viewer Exemplo 1 </h2>

    <applet
     archive="viewer.jar"
     codebase=".."
     code=viewer.ui.Viewer.class
     width=625 height=425>

      <param name="minscale"   value="0.5">
      <param name="maxscale"   value="65536">
      <param name="bgcolor"    value="000000">

      <param name="base"
           value="
           name    'layer'
           layers   1">

      <param name="layer 0"
           value="viewer.layers.Raster
           name    'Shaded Relief (USGS)'
           on
```

```
        image    NorthCoast/nc.jpg
         scale    640
         projection '+proj=utm'
         bounds   '378322 4651998, 704082 4081757'
         overview">

    </applet>

</body>
</html>
```

A Fig. 4 é um exemplo do GIS.4.0, mostrando o mapa da América do Norte, e um detalhe na costa leste dos Estados Unidos. À esquerda se encontra uma lista dos planos de informação disponíveis. Os planos em azul estão habilitados e os em branco, desabilitados. Na parte superior da janela, está a barra de ferramentas com os menus. No canto inferior direito se encontra a barra de ampliação ou mudança de escala. O GIS-4.0 é uma ferramenta fácil de usar e configurar. Como os outros aplicativos apresentados, ele também aceita bases cartográficas no formato SHAPEFILE.

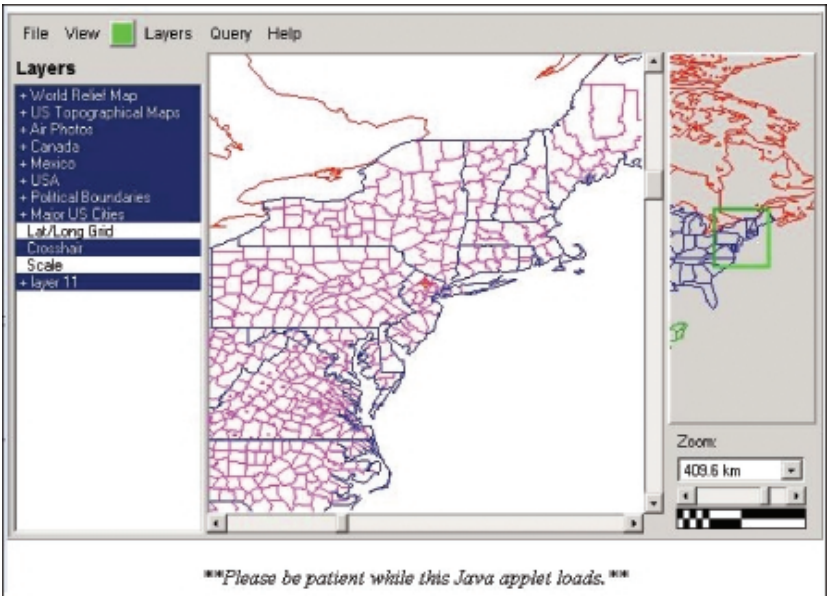


Fig. 4. Exemplo de saída do applet Berkeley GIS4.0.

Alguma limitação é encontrada no que diz respeito a formato vetorial. Para o GIS-4.0, os vetores são todos de uma só cor. Um arquivo vetorial com vários polígonos, como um mapa de solos, não pode ter seus polígonos com cores diferentes. Um arquivo vetorial só pode ser mostrado se todos os seus pares (x, y) forem colocados em um arquivo texto. Embora o programa apresente esta deficiência, ele disponibiliza um grande número de opções para manuseio de mapas e imagens de satélite.

Conclusões

Os dois aplicativos com arquitetura cliente/servidor apresentados implementam a especificação WMS, sendo portanto equivalentes em termos de funcionalidades. Sua grande diferença deve-se ao modo de implementação, Java e C++/CGI. Uma implementação Java é sempre considerada mais segura, do ponto de vista da Internet, pois foi uma linguagem de programação feita com direcionamento para este tipo de aplicação. Ambas implementações permitem o acesso a diferentes sistemas gerenciadores de bases de dados, padrão ANSI SQL, permitindo o manejo de grande quantidade de informações espaciais. A única limitação de ambos se refere à interatividade do usuário com os mapas. Qualquer tipo de interação do usuário com os mapas sempre acontece obedecendo ao ciclo requisição-resposta. Se o usuário usar linha discada, o tempo de resposta pode demorar a cada consulta realizada.

A aplicação cliente, com o GIS-4.0, apresenta grande interatividade do usuário com os mapas, mas não é uma aplicação indicada para se trabalhar com grandes bases de dados por tratar-se de um *applet*. A transmissão de grandes bases de dados é um empecilho para este tipo de aplicação. Portanto, este tipo de aplicação é recomendada se a quantidade de mapas que o usuário quer disponibilizar é pequena. Para aplicações que manipulem grande quantidade de mapas, o mais certo é usar uma implementação do tipo cliente/servidor, como as duas apresentadas, ALOV e MapServer.

Em termos de uso, a versão *servlet* do ALOV é mais fácil do que o CGI do MapServer. Os arquivos de configuração do ALOV são bem mais simples de implementar que o MapServer. Contudo, fica a critério do usuário estudar cada um e fazer sua própria decisão. Além destas opções gratuitas, existem ainda os aplicativos comerciais para publicar mapas na Web.

Referências Bibliográficas

BUEHLER, K. (Ed.) OpenGIS reference model: version 0.1.2. [Wayland]: Open GIS Consortium, 2003. 92 p. (OGC Reference, 03-040).

HUNTER, J.; CRAWFORD, W. Java servlet programming. Sebastopol, CA: O'Reilly, 1998. 510 p.

KOLODZIEJ, K. (Ed.). OpenGIS web map server cookbook: version 1.0.0. [Wayland]: Open GIS Consortium, 2003. 165 p. (OGC Document, 03-050).

MIRANDA, J. I. Diretivas para disponibilizar mapas na Internet. Campinas: Embrapa Informática Agropecuária, 2002a. (Embrapa Informática Agropecuária. Documentos). No prelo.

MIRANDA, J. I. Servidor de mapas para Web: aplicação cliente com o ALOV Map. Campinas: Embrapa Informática Agropecuária, 2002b. (Embrapa Informática Agropecuária. Documentos). No prelo.

NIEMEYER, P.; PECK, J. Exploring Java. Cambridge, MA: O'Reilly, 1997. 594 p.

WALNUM, C. Java em exemplos. São Paulo: Makron Books, 1997. 610 p.

WIRELESS APPLICATION PROTOCOL FORUM LTD. Open Mobile Alliance Ltd. Disponível em: <http://www.wapforum.org>. Acesso em: 6 set. 2002.

Ministério da Agricultura, Pecuária e Abastecimento

Governo Federal