Google Maps e Ajax
Mini-Curso

Prof. Sylvio Silveira Santos
PUC Minas – Betim
Departamento de Sistemas de Informação
Outubro, 2007

Introdução

- Neste mini-curso veremos como passar de alguns conceitos básicos de Ajax e da API do Google Maps à elaboração de Mapas de grande valor estratégico e comercial.

- A base de tudo é Ajax e algumas de suas variações, sobretudo adaptações e mesclagens conhecidas como MASHUPS

- Baseamos estas notas no texto "**Beginning Google Maps Applications With PHP and Ajax**", de Michael Purvis, Jeffrey Sambells e Cameron Turner - Editora Apress, 2007 – USA.

Parte I

- Utilizando a Google Maps API
  - KML – Keyhole Markup Language: um Primeiro Mapa
  - Wayfaring: Segundo Mapa
    - Acrescentando ponto de referência no Mapa
    - Outras possibilidades:
      - Acrescentar uma rota de vôo
      - Marcar o ponto de destino
      - Indicar uma rota terrestre

KML - Primeiro Mapa

- A Keyhole Markup Language é uma linguagem de marcação diferenciada
- Em junho de 2006 a Google anunciou que seu site oficial de mapas iria suportar arquivos em KML - Keyhole Markup Language
- Para compreender o poder desta tecnologia, examinemos o trecho de código a seguir.

Listagem 1.1 - Exemplo de um Arquivo KML

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.google.com/earth/kml/2">
<Document>
<name>toronto.kml</name>
<Placemark>
<name>CN Tower</name>
<description>The CN Tower (Canada's National Tower, Canadian National
Tower),
at 553.33 metres (1,815 ft., 5 inches) is the tallest freestanding structure on
land.
It is located in the city of Toronto, Ontario, Canada, and is considered the
signature icon of the city. The CN Tower attracts close to two million visitors
annually.
http://en.wikipedia.org/wiki/CN_Tower</description>
<Point>
<coordinates>-79.386864,43.642426</coordinates>
</Point>
</Placemark>
</Document>
</kml>
```

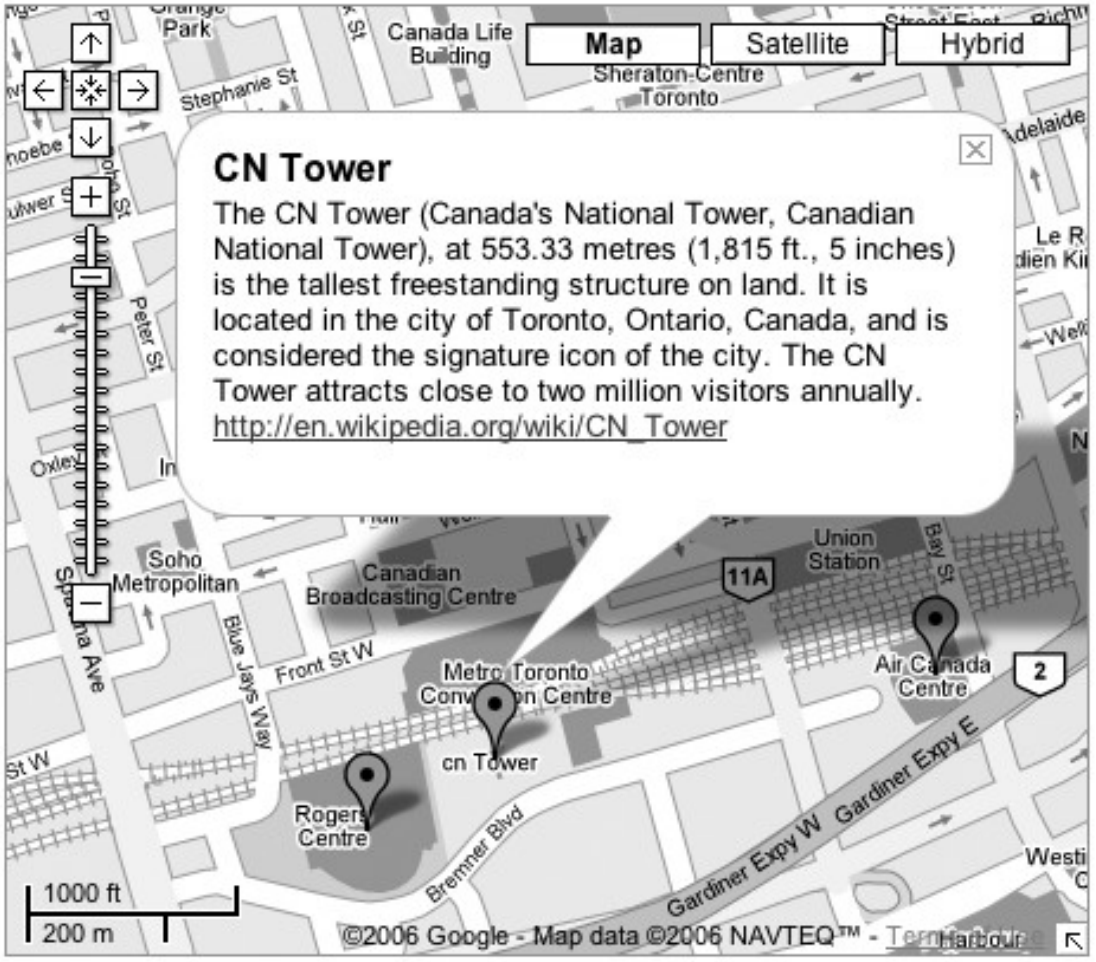
Resultado

Maps Print Email Link to this page

Displaying content from www.googlemapsbook.com
View on Google Earth

The content overlaid onto this map is provided by a third party, and Google is not responsible for it.

- CN Tower
- Rogers Centre
- Air Canada Centre



Wayfaring - Segundo Mapa

- Wayfaring.com é uma aplicação Web 2.0 interativa
- Foi desenvolvida usando o framework Ruby on Rails
- Através da API do Google Maps, permite aos usuários criarem seus próprios mapas com algumas variações
- Endereço: www.wayfaring.com

Determinação da Latitude e Longitude de um Lugar

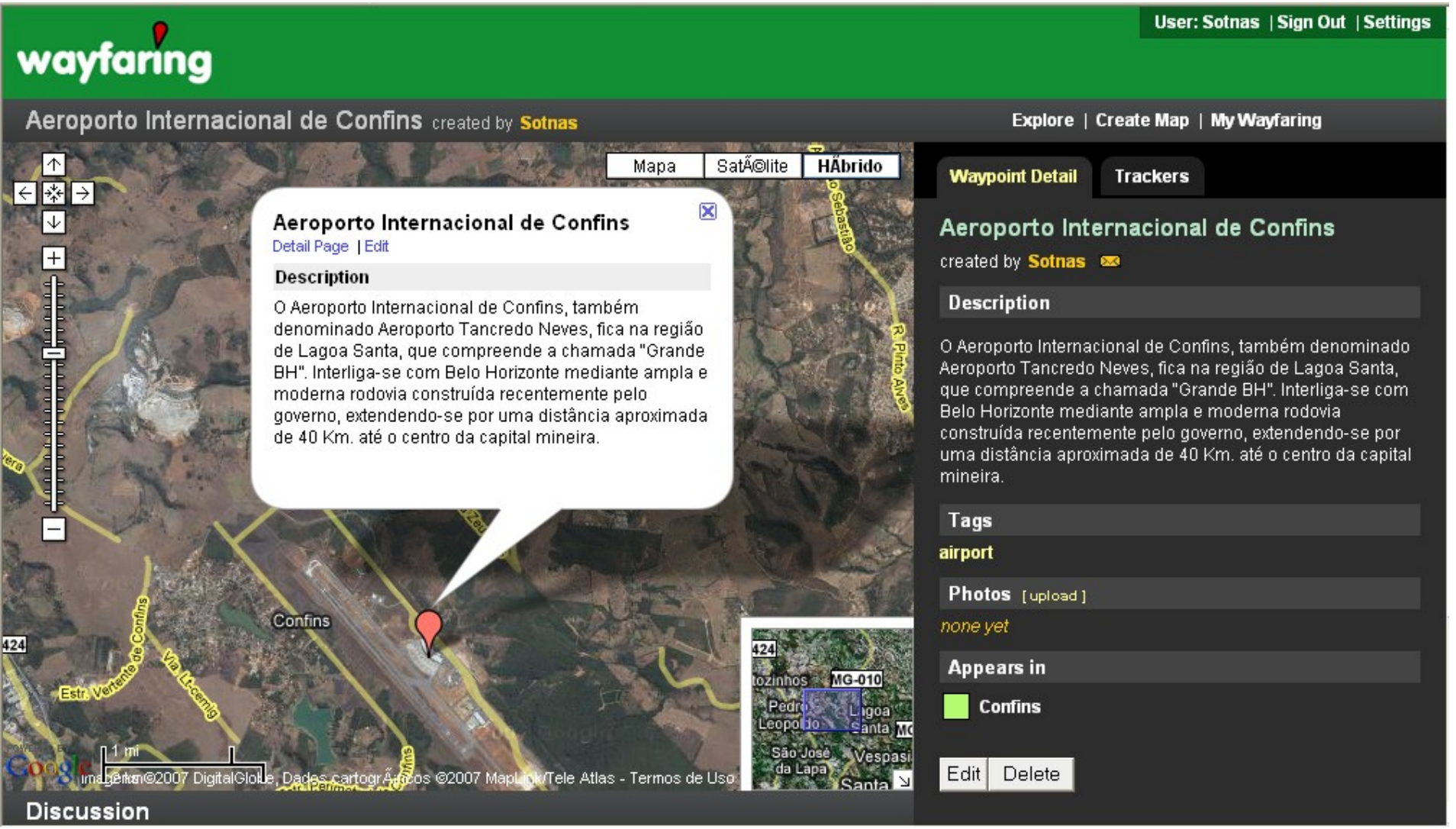
- Direcionemos o navegador para http://mapki.com/getLonLat.php
- Nesta tela mapa nós podemos ajustar a posição de modo a achar a Latitude e a Longitude de um ponto qualquer.
- Entremos com estes valores no Wayfaring, testando, por exemplo, o Aeroporto de Confins.
- Indo para lá podemos ler as coordenadas:
  - Latitude: -19.632603975587745
  - Longitude: -43.96346569061279
- No Wayfaring, digitemos estas coordenadas e clicar em NEXT

Continuando...

- No campo indicado, digitemos o nome Confins e prossigamos...
- Selecionemos Add a Way Point e digitemos:

  “Aeroporto Internacional de Confins”
- Continuando, podemos selecionar um ícone (tag) para marcar a localidade.
- Após isto, digitemos um texto relativo à esta localidade e salvemos o trabalho.

Resultado, clicando na “gota” indicativa da localidade determinada



Extensão do trabalho

- Depois de visualizar parte do trabalho, nós poderiamos retornar à página inicial do Wayfaring e traçar uma rota, por exemplo, do aeroporto até nossa casa.
- Como se nota, o aplicativo Wayfaring possui as características habituais da Web 2 - é interativo e SOCIAL.
- Um aplicativo semelhante e que faço uso com freqüência entre os diversos que podemos encontrar nesta área é o Apontador, em:

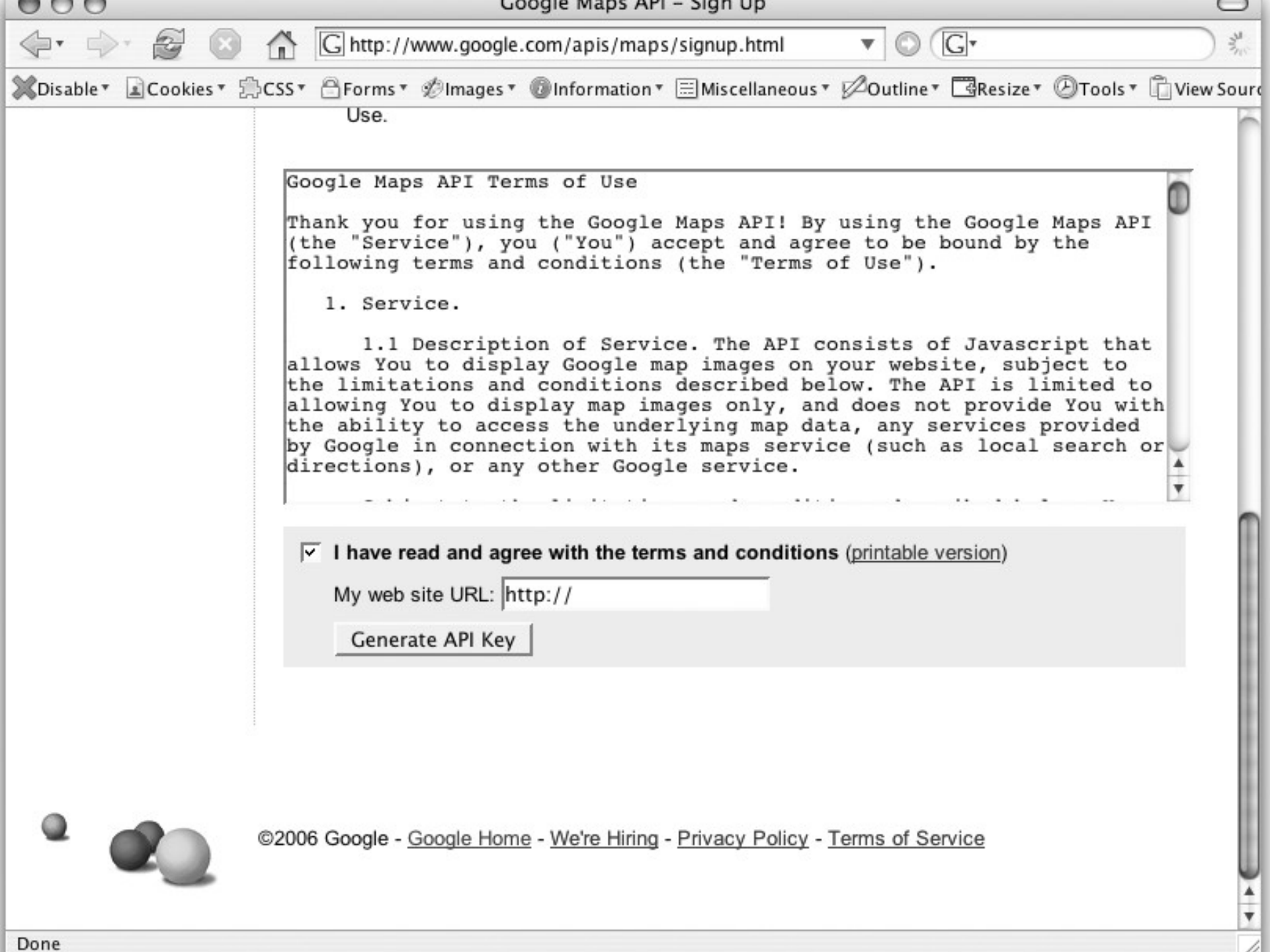
  www.apontador.com.br

Parte 2 - Nosso Primeiro Mapa

- Para iniciar um trabalho com a API do Google Maps, apontemos o navegador para:

  http://www.google.com/apis/maps/signup.html
- A tela da próxima transparência aparecerá.
- No campo apropriado poderemos informar o endereço de um nosso site, ou até mesmo do local onde se hospeda a Home Page deste evento,

  http://www.betim.pucminas.br/si/10anos/programacao.htm
- Dando aceitação aos termos do contrato, enviemos o formulário à Google, a fim de obter a chave.
- Esta chave será parte do código em JavaScript a ser usado mais adiante.
- Salvemos a chave no Notepad


Google Maps API - Sign Up
http://www.google.com/apis/maps/signup.html
Disable
Cookies
CSS
Forms
Images
Information
Miscellaneous
Outline
Resize
Tools
View Sourc
Use.
Google Maps API Terms of Use
Thank you for using the Google Maps API! By using the Google Maps API (the "Service"), you ("You") accept and agree to be bound by the following terms and conditions (the "Terms of Use").
1. Service.
1.1 Description of Service. The API consists of Javascript that allows You to display Google map images on your website, subject to the limitations and conditions described below. The API is limited to allowing You to display map images only, and does not provide You with the ability to access the underlying map data, any services provided by Google in connection with its maps service (such as local search or directions), or any other Google service.
I have read and agree with the terms and conditions (printable version)
My web site URL: http://
Generate API Key
©2006 Google - Google Home - We're Hiring - Privacy Policy - Terms of Service
Done

Código de um Primeiro Mapa

- Uma vez que tenhamos obtido acesso à chave, teremos acesso a um código em JavaScript experimental.
- Copiemos e colemos este código, por meio do Notepad, com a denominação index.html, em um diretório temporário:

  C:/Temp/seu_nome
- Abramos o Navegador e cliquemos no arquivo.
- Maravilha! Nosso primeiro mapa!

Código Gerado - I

- O código que foi gerado durante o registro é uma página Web em XHTML padrão.
- Muitas instruções escritas aqui são padronizadas e se destinam apenas à inicialização do navegador.
- Entretanto existem alguns elementos a considerar:
- Em 1º. Lugar, o **HEAD** do documento possui um script que é crítico. Seu atributo **SRC** aponta para onde está o servidor da Google, e sua chave é repassada como parâmetro:

```
<script src=
    http://maps.google.com/maps?file=api&amp;amp;v=2&amp;amp;ke
    Sua chave aqui    type="text/javascript">
</script>
```

Código Gerado

- Podemos economizar tempo indo diretamente ao código gerado. Verifique no site do livro de Michael Purvis
- Para isto abramos uma sessão paralela do navegador
- Digitemos: www.googlemapsbook.com
- Procuremos por **Listing 2-1 The Google Maps API Starter Code.** Teremos:

```
<?php include $_SERVER['DOCUMENT_ROOT'] . '/apikey.php'; ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8"/>
    <title>Google Maps JavaScript API Example</title>
    <script src="http://maps.google.com/maps?file=api&amp;v=2&amp;key=<?= $api_key ?
    type="text/javascript"></script>
    <script type="text/javascript">

    //<![CDATA[

    function load() {
    if (GBrowserIsCompatible()) {
        var map = new GMap2(document.getElementById("map"));
        map.setCenter(new GLatLng(37.4419, -122.1419), 13);
    }
    }

    //]]>
    </script>
</head>
<body onload="load()" onunload="GUnload()">
    <div id="map" style="width: 500px; height: 300px"></div>
</body>
</html>
```

Discussão do Código - I

- Copiemos este código no Notepad
- Na listagem, o *container* que contém o mapa é uma página web em HTML bastante comum
- Muito do código nela contido é padrão e serve apenas para inicialização do navegador, mas temos alguns elementos importantes a considerar

Discussão do Código - I

- Observemos a seção **Head** do documento. Ele contém um script crítico. Seu atributo **SRC** aponta para a localidade da API do servidor da Google, e nossa chave é passada como parâmetro:
- <script src= http://maps.google.com/maps?file=api&v=2& **NOSSA CHAVE AQUI!** type="text/javascript"></script>

Discussão do Código - II

- Em segundo lugar, a seção **<BODY>** do documento contem uma tag **<DIV>** chamada **map**:

```
<div id="map" style="width: 500px; height:
    300px"></div>
```

- Embora pareça vazia, esta é a parte do documento onde o mapa irá ficar.
- Observemos o atributo de estilo, 500 pixels de largura e 300 pixels de altura.

Discussão do Código - III

- Por fim, voltando ao cabeçalho, existe um elemento de script contendo um pequeno código em JavaScript, o qual é disparado pelo evento **onload**, mais abaixo.
  É este código que se comunica com a API do Google, para geração do mapa:

```
function load() {
      if (GBrowserIsCompatible()) {
      var map = new
GMap2(document.getElementById("map"));
map.setCenter(new GLatLng(37.4419, -122.1419), 13);
       }
 }
```

Discussão do Código - IV

- Observemos que a 1ª. Linha é uma declaração **IF**, que faz a verificação se o navegador do usuário suporta o GoogleMaps.
- A seguir está uma declaração que cria um objeto **GMap2**, um dos muitos objetos importantes que compõem a API.
- O objeto GMap2 é instruído a se acoplar ao mapa identificado em **div**, e fica então associado a uma variável chamada **map**.

OBS Sobre Outros Objetos e Outras Localidades

- OBS 1: Anteriormente nós já nos deparamos com um outro objeto importante: **GLatLng**.
- OBS 2: Como já possuimos as coordenadas do Aeroporto de Confins, experimentemos utilizá-las como abaixo:

```
map.setCenter(new GLatLng(-19.632603975587745,
    -43.96346569061279), 13);
```

Separando Código do Conteúdo

- Vamos procurar separar nosso código em partes distintas:
  - index.php: para a parte em XHTML
  - map_functions.js: para a parte em JavaScript
  - map_data.php para os dados das coordenadas plotadas no mapa
- As listagens a seguir nos mostram esta separação, com dois scripts extras dentro do **HEAD** do documento:

Arquivo index.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"↪
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<script
    src="http://maps.google.com/maps?file=api&amp;v=2&amp;key=
    ↪
ABQIAAAAfAb2RNhzPaf0W1mtifapBRI9caN7296ZHDcvjSpGbL7P
    xwkwBS↪
ZidcfOwy4q2EZpjEJx3rc4Lt5Kg" type="text/javascript"></script>
<script src="map_data.php" type="text/javascript"></script>
<script src="map_functions.js" type="text/javascript"></script>
</head>
<body>
<div id="map" style="width: 500px; height: 300px"></div>
</body>
</html>
```

Arquivo map_data.php

- Observemos que a listagem é a mesma do documento em HTML listado na transparência 17, mas agora existem dois scripts extras dentro do **HEAD**.
- Em vez de se referirem à API, estes scripts se referem aos arquivos em JavaScript denominados **map_data.php** e **map_functions.js**
- Deixaremos por enquanto o arquivo referenciado por **map_data.ph** vazio.

Arquivo map_functions.js

```
var centerLatitude = 37.818361;
var centerLongitude = -122.478032;
var startZoom = 13;
var map;
function init()
{
if (GBrowserIsCompatible()) {
map = new GMap2(document.getElementById("map"));
var location = new GLatLng(centerLatitude,
centerLongitude);
map.setCenter(location, startZoom);
}
}
window.onload = init;
```

Comentários

- O comportamento do código é o mesmo
- A Latitude, Longitude e Altitude do ponto de partida do mapa estão armazenadas em variáveis VAR no alto do script, facilitando alteraçõs destes pontos, se assim desejarmos – evita-se assim uma chamada setCenter( ) perdida no meio do código
- O código de inicialização em JavaScript foi retirado do interior do código em XHTML e passou para o arquivo map_functions.js.
- Podemos assim anexar uma função à chamada de evento window.onload
- O artifício da separação de código permite portanto que se trabalhe independentemente com os arquivos map_functions.js e map_data.php.

Eliminando Lixo em JavaScript

- O lixo que se acumula em uma sessão JavaScript pode ser eliminado com uma instrução simples:

```
window.onunload = GUnload;
```

- Os browsers Firefox e Safari procuram eliminar este lixo deixado, mas o Internet Explorer até versão 6 não o fazia.
- Assim, podemos anexar esta instrução ao código anterior (próxima transparência)

Alguns Widgets para Controle do Mapa

- A API fornece 5 controles padronizados para serem anexados a qualquer mapa:
  - GLargeMapControl, Controle de aumento e diminuição de zoom, o qual é usado em maps.google.com
  - GSmallMapControl, um “mini pan” e controle de zoom, apropriado para mapas pequenos
  - GScaleControl, controle que mostra a Régua de Escalas no mapacenter
  - GSmallZoomControl, controle de zoom de dois botões, usado para indicar direções (pop-ups)
  - GMapTypeControl, que nos permite alternar entre as visões Map, Satellite e Hybrid

Onde colocar estes Widgets

- Em todos os casos, deve-se instanciar o objeto de controle e adicioná-lo ao código através do método addControl que atua sobre o objeto GMap2 com a instrução:

```
map.addControl(new GSmallMapControl( ) );
```

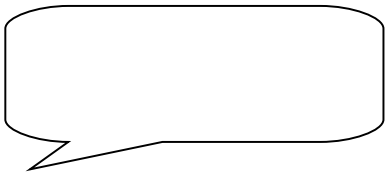
- Podemos usar idênticos processos para adicionar todos os controles, simplesmente passando-os para uma nova instância da classe de controle.

Adicionando Marcadores (Pinos) e Balões

- Marcadores (Pinos) podem ser adicionados ao mapa por meio do método **addOverlay( )**.
- Este método é usado tanto para os pinos como para os “balões”
- Neste caso vamos precisar de:
  - Um objeto GlatLng para armazenar a posição
  - Um objeto opcional GIcon para armazenar a imagem que represente visualmente o marcador no mapa
  - Um objeto GMarker, que é o próprio marcador
  - Um objeto GMap2 terá o marcador plotado nele, por meio do método addOverlay( )

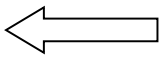
pino

Balão



Código Com Pino

```
var centerLatitude = 37.818361;
var centerLongitude = -122.478032;
var startZoom = 13;
var map;
function init()
{
if (GBrowserIsCompatible()) {
map = new GMap2(document.getElementById("map"));
map.addControl(new GSmallMapControl());
var location = new GLatLng(centerLatitude, centerLongitude);
map.setCenter(location, startZoom);
        var marker = new GMarker(location)
        map.addOverlay(marker);
}
}
window.onload = init;
window.onunload = GUnload;
```



Coloque setCenter antes do marcador

Código com Pino Personalizado

- Se tivermos um ícone personalizado, como o símbolo de um aeroporto, podemos colocá-lo no mapa mediante a instrução:

  var marker = new GMarker(**minha_GLatLng**, **meu_GIcon**);
- Notemos que quando **Icon** não é especificado, a API supõe que a gota vermelha invertida é usada como *default*.

Eliminando lixo

```
var centerLatitude = 37.818361;
var centerLongitude = -122.478032;
var startZoom = 13;
var map;
function init()
{
if (GBrowserIsCompatible()) {
map = new GMap2(document.getElementById("map"));
var location = new GLatLng(centerLatitude,
centerLongitude);
map.setCenter(location, startZoom);
}
}
window.onload = init;
window.onunload = GUnload;
```

Detecção de Cliks sobre o Marcador - I

- Admitamos ter no código um objeto GMarker chamado marker
- Vejamos o código:

```
function handleMarkerClick() {
alert(“Você clicou no marcador!");
   }
GEvent.addListener(marker, 'click',
 handleMarkerClick);
```

Detecção de Cliks sobre o Marcador - II

- Entretanto isto não funciona quando se tem vários marcadores.
- Podemos então passar function diretamente como parâmetro em addListener( ):

```
GEvent.addListener(marker, 'click',
function() {
        alert(“Você clicou no marcador!");
    }
 );
```

Método openInfoWindowHtml()

- Podemos fazer o método openInfoWindowHtml() atuar sobre um marcador, marker, e com isto abrir uma descrição nele. Vejamos:

```
GEvent.addListener(marker, 'click',
 function() {
marker.openInfoWindowHtml(description);
                }
 );
```

Código Gerando Pino, Evento e Info Window - I

```
var centerLatitude = 37.818361;
var centerLongitude = -122.478032;
var description = ‘Ponte Golden Gate';
var startZoom = 13;
var map;
function addMarker(latitude, longitude, description) {
var marker = new GMarker(new GLatLng(latitude, longitude));
GEvent.addListener(marker, 'click',
function() {
marker.openInfoWindowHtml(description);
}
);
map.addOverlay(marker);
}
function init() {
if (GBrowserIsCompatible()) {
map = new GMap2(document.getElementById("map"));
map.addControl(new GSmallMapControl());
map.setCenter(new GLatLng(centerLatitude, centerLongitude), startZoom);
addMarker(centerLatitude, centerLongitude, description);
}
}
window.onload = init;
window.onunload = GUnload;
```

Estrutura de Dados em JavaScript – Lista de Localidades

```
var markers = [
{
'latitude': 37.818361,
'longitude': -122.478032,
'name': 'Golden Gate Bridge'
},
{
'latitude': 40.6897,
'longitude': -74.0446,
'name': 'Statue of Liberty'
},
{
'latitude': 38.889166,
'longitude': -77.035307,
'name': 'Washington Monument'
}
];
```

Ressalva Sobre Base de Dados

- O uso de mySQL é indicado para armazenar um grande número de dados
- Uma alternativa a SQL é JSON (JavaScript Object Notation)
- Podemos accessar o site de JSON em www.json.org

Interação em JavaScript

- Podemos criar um loop para passar um bloco reprtitivo de código usando interações:

```
for (id = 0; id < markers.length; id++) {
// criar um marcador em markers[id].latitude, markers[id].longitude
}
```

- Uma solução mais elegante seria dada por:

```
for (id in markers) {
// criar um marcador em markers[id].latitude, markers[id].longitude
}
```

Chamada de Marcadores

- Devemos notar que se pode usar o for no arquivo map_functions.js para chamar marcadores de map_data.php

- No código que se segue, a função addMarker foi chamada para cada um dos marcadores em três janelas de informação diferentes.

```
var map;
var centerLatitude = -95.0446;
var centerLongitude = 40.6897;
var startZoom = 3;
function addMarker(longitude, latitude, description) {
var marker = new GMarker(new GLatLng(latitude, longitude));
GEvent.addListener(marker, 'click',
function() {
marker.openInfoWindowHtml(description);
}
);
map.addOverlay(marker);
}
function init() {
if (GBrowserIsCompatible()) {
map = new GMap2(document.getElementById("map"));
map.addControl(new GSmallMapControl());
map.setCenter(new GLatLng(centerLatitude, centerLongitude), startZoom);
for(id in markers) {
addMarker(markers[id].latitude, markers[id].longitude, markers[id].name);
}
}  }
window.onload = init;
window.onunload = GUnload;
```

Trabalhos de Meus Alunos no Barreiro - I

- Bares e Restaurantes nas proximidades da PUC Minas - Barreiro

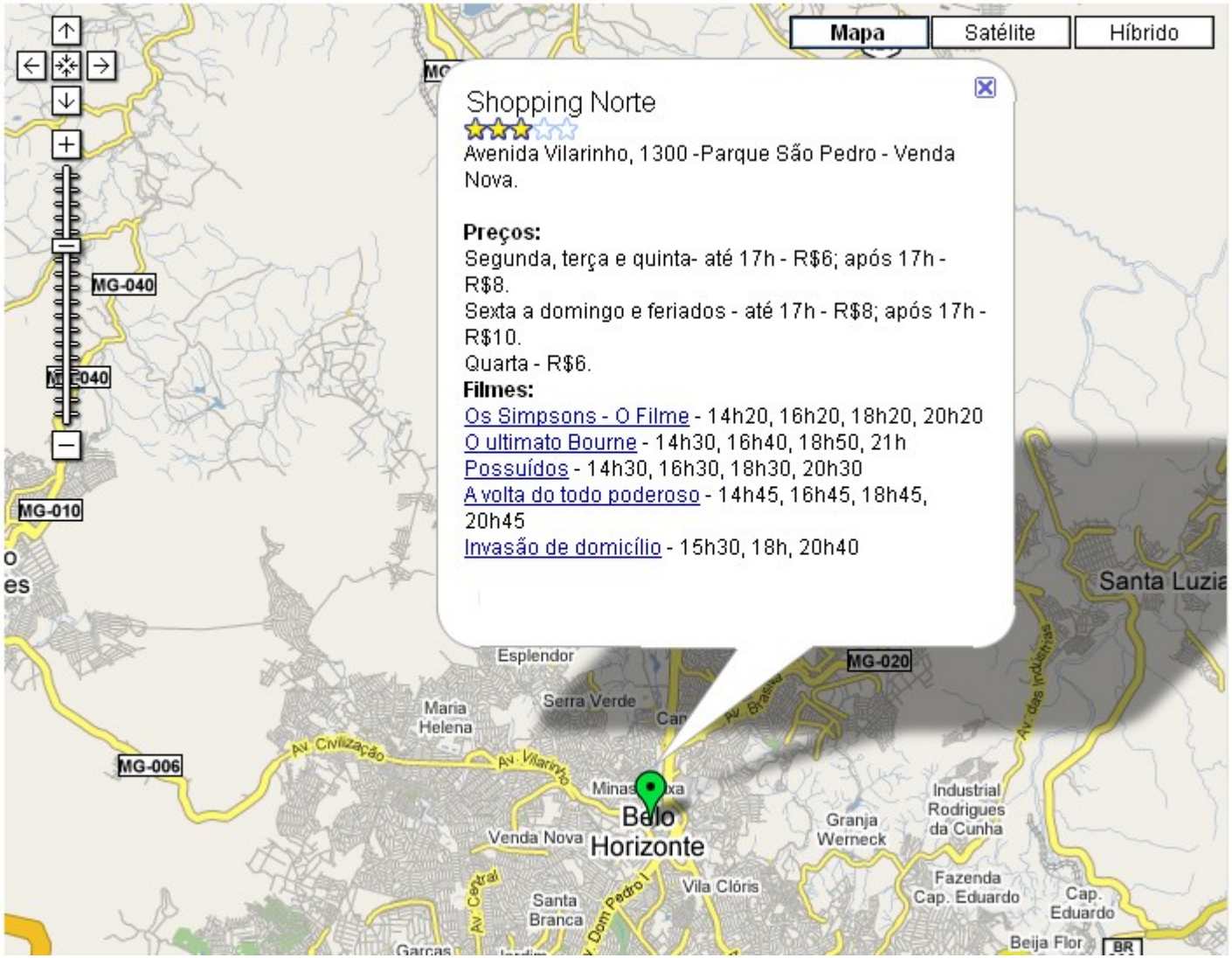
Alunos:

- **Adriano de Araújo**
- **Cristiano Eduardo D'Alessandro**
- **Leandro Ribeiro Matoso**
- **Rômulo Amâncio Vieira**

Em http://www.divirtaseembh.com.br/mapas/

Trabalhos de Meus Alunos no Barreiro - II

Encontre cinemas em BH: http://cinemasbh.googlemashups.com



Leituras e Sites Recomendados

- http://googlemapsbook.com
- http://googlemapsbook.com/appendixa/links.html
- www.programmableweb.com
- www.google.com/apis/maps/documentation/reference.html
- www.nuforc.org
- Zammetti, Frank, Ajax Projects with Java Technology – Apress, USA, 2007
- Mahemoff, Michael, Ajax Design Patterns – Creating Web 2.0 Sites with Programming Usability and Patterns – O'Reilly, USA, 2006
  Obs: Existe tradução do livro de Mahemoff em Português