IGT-Site, a portable, database driven Web Site

by
P. Fritz
Dr. sc. techn.
Division of Geotechnical Engineering (http://www.igt.ethz.ch/)
Swiss Federal Institute of Technology, Zürich

Excerpt of a PowerPoint Slide Show
to demonstrate design and implementation of the IGT-Web Site

The impulse to developing a portable Web site was the immediate need of the Division of Geotechnical Engineering to overhaul its four years old Web site. But as expected from a research organization we did not want to produce just some pages for the public, but we took the opportunity to investigate the various approaches for building a Web site from a more general point of view.

The result is a fairly general Web site which is portable and may be used by a variety of organizations. It is tightly connected to a database, where most of the content of the Web site is stored. Inputting data to the Web site may be done also by untrained personnel thanks to an easy-to-handle form based user interface. This has the advantage that the work of updating the content of the Web site is not concentrated in a single person, usually the Webmaster, but may be spread to practically all members of the organization, each of which may be made responsible for that content he is responsible for or interested in also during his professional work. It is expected that due to this concept the content of the Web site should always be up-to-date.

From a technical point of view the Web site is based on Microsoft’s Active Server Pages, which run on an NT-Webserver and produce pure HTML code which is sent to the browser. With this approach firstly no additional software is needed on the client side, and secondly basically all browsers running under any operating system are supported.
IGT-Site, a portable database driven Web Site

- Motivation
- Aim
- Design
- User Interface
- Implementation
- Outlook
- Summary
Motivation

ETH internally
New politics at ETH

Expenditure for old Site
Great effort since 1994, hard to update

Offering more Content
Full text of Publications
People

IGT-Site, a portable DB driven Web Site

Four years ago, running one’s own Web site was for a company a means for showing its open-mindedness towards modern computer aids. Today it is a desirable (and cheap) component in communicating its services to a broader public, and in the near future it will be as natural and common as having a calling card. As the novelty of merely having a Web presence wears off, companies are beginning to focus their attention on building useful Web sites, i.e. publishing their data and services.

However our experience with running a Web site over the last four years revealed that the maintenance of a larger Web site may be extremely demanding and very time-consuming. Firstly, because keeping the content up-to-date not only means entering the latest changes, but also that somebody is required who intimately knows the company’s organization and who is constantly chasing after its staff about any possible changes and extensions of its data.

The IGT-Site was developed as a common effort of the Division of Geotechnical Engineering (IGT) and an external company (EBSL). The former has been responsible for the concept and design, the latter for the main part of the implementation. Such a collaborative work may give rise to potential hazards: the designer may not consider the possibilities and limits of an implementation, and the programmer may not make allowance for the requests of the designer. As an example of this dilemma the reaction of the Webmaster of the Swiss Federal Institute of Technology (ETH) may serve, which after introducing the new Web site of ETH complained, that he could not take any influence once the project was started ("Natürlich habe ich schon während der Design-Phase versucht Einfluss zu nehmen, leider vergeblich" (Ambühler, 1997). We avoided this problem by a mutual agreement, which states that the design and development process had to be carried out in an iterative way by both parties, but that the copyright belonged solely to IGT.

To comply with and take advantage of the higher level rules of the ETH, this site has been designed in accordance to the "Rules for and Recommendations to the Web Politics of ETH Zürich" (WWW-Kommission, 1997).
IGT-Site, a portable DB driven Web Site

Motivation
Aim
Design
User Interface
Implementation
Outlook
Summary
A basic aim of the IGT-Site was to guarantee the freshness and completeness of its content with minimal administration expense during its whole lifetime. The fundamental design decision to fulfill this aim was to delegate inputting and updating the content of the Web site to the company members themselves. Because of the inherent hazards of such a concept with regard to the correctness and consistency of the data, a specific security disposition had to be developed.

Possibly the most useful applications of a Web site are to publish data to its customers, to serve as a forum for internal information, and to save employees’ time when looking for business related information on the Web.

- The first and foremost function of a Web site is to publish a company’s data to its customers. For an academic institution this means making available data about its staff, its publications and its projects. As further topics it should also provide information about lectures and courses, as well as services offered to other research groups and to the practice. Furthermore it should be prepared to include additional information when it is recognized to be important. An example of this are important links to other Web sites related to the company’s fields of interest.

- For members of the company the IGT-Site should also provide a forum for publishing internal information. An example of this is a guide on how to add to and change the data of the Web site.

- In the last few years the Internet has grown to such an extent that casual users may lose much time just for learning what services they may expect from the Internet, not to mention how to find them. To save precious working time a guide should be provided which helps company members to find information related to their work.

Besides the content of a Web site, the way of presentation and the user interface are most important for its attractiveness. We tried to fulfill these aims with the following concept:

- navigation: the navigation scheme should be consistent and simple.
- cross references: related information should always be just one mouse click away.
- consistency: the whole Web site has to exhibit a consistent look and feel.
- focal point: the presentation should be elegant, but concentrating on valuable content, i.e. refraining from eye-catching features.
IGT-Site, a portable DB driven Web Site

Motivation
Aim
Design
User Interface
Implementation
Outlook
Summary
Design of IGT-Site

Basics
Dynamic or live data, DB, browsers, scalability

Database Connectivity
Client or server control, CSS, ASP

Why using a DB?
Update, user level, consistency of data and look & feel

Access to the DB
Browser via ASP and ADO, Frontend in MS-Access
Dynamic data are generated by fetching data from a server-sided database and sending them to the client’s browser. Important issues of this concept are where to store the data, how to update them and how to make them available to Web browsers. As the IGT-Site heavily relies on dynamic data, these issues will be discussed in more detail below.

Manipulating live data refers to client-sided processing of a disconnected data set. A disconnected data set outlives the connection to the database server. This is advantageous if the data need to be manipulated heavily without updating them from the server. A disadvantage of this concept is that the programming model for getting and manipulating the data is proprietary. Microsoft e.g. offers the Remote Data Service (RDS) programming model, which is only supported by Microsoft (MS) Internet Explorer. The IGT-Site does not manipulate live data.

The design of a Web site may vary very much depending of the expected traffic volume. The IGT-Site is certainly not conceived for very high traffic demands such as occur on important sites with thousands or millions of accesses per day. However several provisions are included which allow to scale the site to cover increasing demands. These will be described under the heading "Outlook".

Unfortunately, the capabilities of the various browsers are quite different. Therefore, a principal design decision is which browsers should be supported, or which features should be used. In this respect the IGT-Site has been designed taking into account the following considerations:

- MS Internet Explorer and Netscape cover about 95% of the market. Both will be supported. Other browsers have not been checked, which however does not mean that they cannot be used.
- The features of version 4 of both browsers have been so much improved compared to their predecessors (e.g. support of frames, Java scripts v.1.1, etc.), that they will fully be exploited. Both browsers are available free of charge, i.e. it may be expected that they will soon be used by the majority of users. Therefore, their predecessors will not be supported.
- To facilitate support, extremely few browser-dependent code is used.
- In view of the ongoing dispute concerning the security problems of Java applets and ActiveX controls, Java applets are not used. ActiveX controls are only used server-sided, thereby guaranteeing their control and avoiding any unwanted side effects.
Design of IGT-Site

Basics
Dynamic or live data, DB, browsers, scalability

Database Connectivity
Client or server control, CSS, ASP

Why using a DB?
Update, user level, consistency of data and look & feel

Access to the DB
Browser via ASP and ADO, Frontend in MS-Access
Models for DB Connectivity

2-tier architecture
- complicated programming
- great SW power
- logic client-sided
- complicated SW distribution
- permanent connection with client-sided transaction support

3-tier architecture
- simple HTML for client, adjustable complexity for middleware
- limited SW power
- logic hierarchically distributed
- simplest SW distribution (none)
- transient connection (data transfer)
  stateless model
  no client-side transaction support

Browser

DBMS

Web Server

Browser
Models for DB Connectivity

Browser → Web Server → DLL → DBMS

CGI

ASP

Swiss Federal Institute of Technology
Models for DB Connectivity

- **Common Gateway Interface**
  - independent programs in separate processes
  - PERL, REXX, or any prog. language

- **proprietary: ISAPI, NSAPI**
  - efficient, same process as server
  - any prog. Language
  - comfortable development environment (FP, Livewire)
  - may perjudice server process

- **Proprietary (see also ColdFusion)**
  - efficient, same process as server
  - scripts (VB, JS), JApplets, ActiveX controls
  - comfortable development environment (Visual IDev)

Comparison to other approaches:

- **ASP** is faster than CGI: ASP provides the flexibility of CGI programs and scripts, without the significant performance tradeoff. Unlike CGI, ASP runs in-process with the server, and is multi-threaded and optimized to handle large numbers of users.

- **ISAPI applications** require all of the programming and layout to be contained in a dynamic link file written in C++ or a similar language. ISAPI applications are thus more difficult to create and maintain. With ASP files, an HTML writer can script an external component and format the output without semantic breaks.

- **Netscape LiveWire** requires the use of *JavaScript*, while ASP support the use of virtually any scripting language, with native support for *VBScript* and *JScript*. ASP supports components written in any language while *LiveWire* supports only Java components.
Design of IGT-Site: Database Connectivity

- Two fundamentally different strategies:
  1. browser-sided plug-ins (2-tier architecture)
  2. server-sided control (3-tier architecture)

When using browser-sided plug-ins these are responsible for connecting to the DB and displaying the output inside the browser. The plug-ins such as ActiveX controls or PowerBuilder objects are usually created with rapid application development (RAD) tools. Because the client accesses the database directly, additionally to the plug-ins also database drivers are required on the client side.
In the approach based on server-sided control, the server itself connects to the DB using any required middleware, as e.g. *Active Server Pages* (ASP), *Java* or *Common Gateway Interface* (CGI)-scripts. It then creates the user interface in standard *Hypertext Markup Language* (HTML), possibly complemented by *JavaScript*, and sends it back to the browser. With this approach the client does not need any additional software.

Our design concept is based on the second approach. More precisely we rely on the *Windows Distributed Internet Application Architecture* (DNA), i.e. client-sided *Java* scripts, *HTML* 4.0 with *Cascading Style Sheets*, server-sided ASP scripts and server-sided includes, and database interrogation by the *Structured Query Language* (SQL). From a software point of view we use NT with the MS *Internet Information Server*.

The main advantage of this approach is that the client does not need to download and install any additional software, which in addition to the aspects of comfort and simplicity avoids the problem of operating system and browser dependency.
Active Server Pages (ASP)

The simplest example:

```html
<HTML>
<BODY>
Hello World !
</BODY>
</HTML>
```

Additionally ASP may contain code (scripts, ActiveX controls) within `<% ... %>` which is interpreted by the server.

run example:
An ASP-based **Application** is defined as all the .asp files in a virtual directory and its subdirectories.

A **Session** object is created when an ASP page from the application is requested by a user who does not already have a session. The server destroys the Session object when the session expires or is abandoned. Variables stored in the Session object are not discarded when the user jumps between pages in the application; instead, these variables persist for the entire user-session.

Session state is only maintained for browsers that support cookies, because the SessionID is stored in a Cookie (variables in server's RAM).

**Active Server Pages** includes a number of "**built-in**" server and application building **objects**. These objects free developers from the grind of writing code to access details about incoming requests from clients, managing the application state, handling cookies, and assembling the response. These intrinsic objects also include **ActiveX Server Components**. One of these is the **ActiveX Data Object** (ADO), an **ActiveX Server Component** which contains the logic to talk to a variety of data stores (e.g. MS Exchange, Active Directory, MS Project) through a common set of interfaces. We use ADO to talk to our relational database.
ASP Session Management

Built in ASP objects (http://www.activeserverpages.com/iishelp/iis/htm/asp/iiwaobb.htm)

<table>
<thead>
<tr>
<th>Object</th>
<th>Collections</th>
<th>Properties</th>
<th>Methods</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Contents</td>
<td>StaticObjects</td>
<td>Lock Unlock</td>
<td>Application_OnStart</td>
</tr>
<tr>
<td></td>
<td>StaticObjects</td>
<td></td>
<td></td>
<td>Application_OnEnd</td>
</tr>
<tr>
<td>Session</td>
<td>Contents</td>
<td>SessionID, Timeout, Codepage</td>
<td>Abandon</td>
<td>Session_OnStart</td>
</tr>
<tr>
<td>Request</td>
<td>Form, QueryString, Cookies, ServerVars</td>
<td>TotalBytes</td>
<td>BinaryRead</td>
<td>Session_OnEnd</td>
</tr>
<tr>
<td>Response</td>
<td>Cookies</td>
<td>Content, Expires, Status, IsClientConnected</td>
<td>BinaryWrite, Clear, Flush, Write, End</td>
<td></td>
</tr>
<tr>
<td>Server</td>
<td></td>
<td>ScriptTimeout</td>
<td>CreateObject, MapPath, HTMLEncode, URLEncode</td>
<td></td>
</tr>
<tr>
<td>ObjectContext</td>
<td></td>
<td></td>
<td>SetComplete, SetAbort</td>
<td></td>
</tr>
</tbody>
</table>

**Application** Object: to share information among all users of an application. Contents contains items added by script commands, StaticObjects by <OBJECT> tags. Lock is necessary to lock properties for multiple access.

**Session** Object: to store information for a particular user session. Variables stored in the Session object are not discarded when the user jumps between pages in the application.

**Request** Object: to gain access to any information that is passed with an HTTP request. This includes parameters passed from an HTML form using either the POST method or the GET method, cookies, and client certificates. The Request object also gives you access to binary data sent to the server, such as file uploads.

**Syntax** is usually: Object[.collection|property|method]. For the Request object it is Request[.collection|property|method](variable)

**Response** Object: to send information to the browser, redirecting the browser to another URL, or setting cookie values.

**Server** Object: provides access to methods and properties on the server. The most frequently used method is the one that creates an instance of an ActiveX component (Server.CreateObject).

**ObjectContext** Object: to commit or abort a transaction initiated by an ASP script.
Thanks to its structured design, several options are available for scaling the *IGT-Site* if speed demands increase. A first possibility is based on the way the database is accessed via the *ActiveX Data Object* (ADO). ADO is Microsoft's strategic, high-level interface to all kinds of data. It provides consistent access to data, whether stored as a database, a business object, or even as an Internet resource.

ADO relies on OLE DB, Microsoft's strategic low-level interface to data. The hierarchy for accessing data via ADO and ODBC is illustrated above. At the moment the only data provider available is the *Open Database Connectivity* (ODBC) engine. In the near future an SQL engine should also be released. Just by exchanging the ODBC provider with the SQL provider the data access would become much faster without changing a single line of the ASP code.
Active Server Pages (ASP)

Connection to a database (DB):

```html
<HTML>
<BODY>
<% Set objConn = Server.CreateObject("ADODB.Connection")
objConn.Open "IGT-DB"
Set objList = objConn.Execute("SELECT * FROM TPersons")

Do while not objList.EOF
<%= objList("PerName") %><BR>
<% objList.MoveNext %>
Loop

objList.Close
objConn.Close %>
</BODY>
</HTML>
```

1. instantiate a server Object for an ADO-DB connection
2. open the connection to the DSN (data source name)
3. issue an SQL command and get results
4. display the results
5. close connection

Use a form and retrieve data

Example:
Active Server Pages (ASP)

Query a DB from a Form

A simple Form:

```html
<HTML>
<BODY>
  <FORM METHOD="POST" ACTION="Sample5.asp">
    1st Names to search for: <INPUT NAME="FirstName">
    <INPUT TYPE=SUBMIT>
  </FORM>
</BODY>
</HTML>
```

QUERY.ASP is called with parameter FirstName

run example:
Active Server Pages (ASP)

Query a DB from a Form

<HTML>
<BODY>
  <% Set objConn = Server.CreateObject("ADODB.Connection")
  objConn.Open "IGT-DB"
  FName = Request("FirstName")
  SQLQuery = "SELECT * FROM Tpersons"
  "WHERE TPersons.PerFirstName='" & FName & "'"
  Set objList = objConn.Execute( SQLQuery )
  Do while not objList.EOF
  <%= objList("PerName") %><BR>
  <%= objList.MoveNext %>
  Loop
  objList.Close
  objConn.Close %>
</BODY>
</HTML>

run example:

1. get parameter from form (Request object)
2. add a WHERE clause with this parameter
Design of IGT-Site: Database Connectivity

- **server-sided ASP scripts:**
  - combines the ease of HTML with familiar tools like *Visual Basic Scripting* and *ActiveX Server Components*
  - ASP also supports any scripting language (e.g. *Visual Basic Scripting Edition* and *MS Jscript*)
  - server-sided processing, standard HTML output
  - browser just sees output, not code
  - includes *ActiveX Server Components* (e.g. the *ActiveX Data Object* (ADO))

- **server-sided includes:**
  - adds text, graphic, or application info to HTML
  - get info about a file or display a CGI variable

ASP is an open, compile-free application environment in which one can combine (server-sided) HTML pages, scripts in any language, and *ActiveX Server Components*. While the *Internet Information Server* (IIS) will continue to support *Common Gateway Interface* (CGI) and *Internet Server Application Interface* (ISAPI) programs for Web-specific applications and filters, ASP offers a powerful component-based approach for applications development with the following advantages:

- It supports existing investments: ASP does not require developers to learn a new environment. It combines the ease of HTML with familiar tools like *Visual Basic Scripting* and *ActiveX Server Components*. For experienced Web developers, ASP also supports any scripting language and components written in any language, including *Java*. Native support is provided for *Visual Basic Scripting Edition* and *Microsoft JScript*. Scripts and components are processed on the server and the output of an ASP file is standard HTML, so ASP work with any Web browser.

- A Web user is not able to view the actual source code for ASP. All the browser sees is the HTML output of the ASP page.

- It is compile free: ASP is integrated into *Windows NT* Server, so it knows when a file has changed. A changed script is automatically compiled the next time it is requested. This means that when developing scripts and applications, one can save the page and immediately preview it in the browser.
Design of IGT-Site: **Database Connectivity**

- **HTML 4.0 (= Dynamic HTML):**
  - Document Object Model
  - Cascading Style Sheets

- **Document Object Model (DOM):**
  - access to all page elements (via ID’s, hierarchical objects)
  - instant page update (scripts)
  - full event model (e.g. OnMouseOver)
  - changing the text on the page (e.g. with InnerText)

- **Cascading Style Sheets:**
  describe how documents are presented (e.g. fonts, spacing, and aural cues). Separation of the presentation style from the content of documents.

HTML 4 relies on the Document Object Model (DOM) which allows to access all page elements, to instantly update them, and to change text on the fly. It supports the full event model and includes Cascading Style Sheets. Dynamic HTML is often used as a synonym for HTML 4.
Design of IGT-Site

Basics
Dynamic or live data, DB, browsers, scalability

Database Connectivity
Client or server control, CSS, ASP

Why using a DB?
Update, user level, consistency of data and look & feel

Access to the DB
Browser via ASP and ADO, Frontend in MS-Access
Design of IGT-Site: Why using a DB?

Reasons to avoid hardcoding all information in HTML-pages:

**Separation of data and presentation.**

**Easier to keep data up to date.**

much easier to delegate responsibility to interested individuals (knowledge level!).

**Information** may be organized and **structured** in sophisticated way, thereby avoiding duplication and redundancy.

**Look and feel** of the entire Web site becomes very consistent

The relational database stores all data specific to the needs of our organization, e.g. data about people, publications, projects, products etc. The reasons not to hardcode all this information in HTML-pages are the following:

- it is very difficult for a Webmaster to keep such data up to date. It is much easier to delegate responsibility to the individual people which also have a direct interest in their data appearing on the Web. The prerequisite of this approach is that these data may be entered by anybody without having special knowledge of HTML or databases.
- the information may be organized and structured in a sophisticated way, thereby avoiding duplication and redundancy.
- the *look and feel* of the entire Web site becomes very consistent
Design of IGT-Site

Basics
Dynamic or live data, DB, browsers, scalability

Database Connectivity
Client or server control, CSS, ASP

Why using a DB?
Update, user level, consistency of data and look & feel

Access to the DB
Browser via ASP and ADO, Frontend in MS-Access
Design of IGT-Site: Access to the DB

- For displaying data from the browser:
  - via ASP and ADO:
    SQL query is packed in a server object, dispatched and parsed and executed by the Web server

- For editing data: from a User Frontend:
  - Easy to use forms in MS-Access.
    (Access, in contrast to HTML, provides support for forms (e.g. relational forms), transactions)

The data store itself may reside in any SQL compliant database. For the sake of simplicity we use MS Access.

Data are accessed either from the browser via ASP, or from a Frontend. ASP programming is done by experts and relies on SQL for getting data from the DB. An SQL query is packed in a server object, parsed by the server, and executed. A performance improvement could be reached by using SQL-stored procedures, which are precompiled and do not have to be parsed each time by the server.

In contrast the Frontend may be used by untrained personnel to enter and edit all data. We built an MS Access Frontend with easy-to-use forms. Thanks to a direct connection to the data store, it is possible to verify the consistency of the input data immediately. The reason for using MS Access and not to program the Frontend in HTML is based on the fact that MS Access provides a multitude of utilities and instruments for this purpose, e.g.

- local input verification: the consistency of the DB may be verified/guaranteed during the input itself;
- better support for forms, e.g. interfaces that change as form fields are filled in, gray out buttons, grouping of associated fields;
- relational forms are not supported in HTML.

This led to a very efficient development of the Frontend. Possibly with the dissemination of HTML version 4 the Frontend could be transferred to HTML, with the (only?) advantage that it would not be tied to MS Windows any more. However, the required expense of several man months may not justify this investment.
IGT-Site, a portable DB driven Web Site

Motivation
Aim
Design
User Interface
Implementation
Outlook
Summary
Web Interface

3 Frame Layout:
- header frame
- table of contents (TOC)
- topic or body frame

The **header frame** is invariable and thus it gives the user the feeling of calm and continuity. In addition, from there the most important topics are always just one mouse click away.

The **TOC frame** serves to display the desired pages by clicking on its items. The TOC enhances in a powerful way the orientation regarding where one currently is on the site. For this purpose it is implemented in a bi-directional form, i.e. when clicking on an icon in the TOC the corresponding page is displayed in the body frame, and conversely, if a page is loaded in the body frame, the TOC is automatically updated to display its location in the Web site structure.

Finally, the actual information is displayed in **the body frame**. Following links from within the body frame leads to an update of the body and the TOC frame (as explained above) if the link points to a page within this Web site. For external links the TOC remains unchanged. However, it is possible for external links to override both the TOC and the body frame, thereby fully exploiting the available space, but still with the reposing pole of the header frame.
The Web browser displays the data of the Web site. To enter the data the MS Access Frontend with its forms may be used. The Fig. shows the first screen where the user can choose the kind of data he wants to add or edit.
When choosing the kind of data to edit some more forms and tabs pop up where the data may be entered. Instead of a help file tooltip help is displayed when moving the mouse over an icon.
The same screen for the administrator contains some more tabs for carrying out administrative work.
## Formats of online Papers

<table>
<thead>
<tr>
<th>Paper Format</th>
<th>Description</th>
<th>Pros.</th>
<th>Cons.</th>
<th>Example</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adobe Portable Document Format</td>
<td>de facto standard for electronic distribution of documents because it is the best way to keep the look and feel of a text intact. PDF files are compact, cross platform and can be viewed by anyone with a free Acrobat Reader.</td>
<td>precise layout, supports formula, graphics, small file size thanks to compression, viewable from a variety of platforms as Windows, Mac, Unix.</td>
<td>browser must be configured with a special plug in, which is available free of charge from Adobe.</td>
<td>example with graphics.</td>
<td>55 kB.</td>
</tr>
<tr>
<td>ASCII text</td>
<td>was the most common format for text files in computers and on the Internet.</td>
<td>supported by all browsers without any additional components.</td>
<td>for plain text only, without formatting, without graphics etc.</td>
<td>same example but without graphics.</td>
<td>3 kB.</td>
</tr>
<tr>
<td>MS Word text</td>
<td>proprietary text editor of Microsoft.</td>
<td>good formatting possibilities.</td>
<td>not portable, large files, only Word-95 supported.</td>
<td>same example with graphics.</td>
<td>721 kB.</td>
</tr>
<tr>
<td>HTML text</td>
<td>the Hypertext Markup Language (HTML), is the standard of the WWW.</td>
<td>supported by all browsers without any additional components.</td>
<td>limited formatting possibilities.</td>
<td>same example with graphics.</td>
<td>58 kB.</td>
</tr>
<tr>
<td>Rich Text Format</td>
<td>export format supported by many text editors.</td>
<td>good formatting possibilities.</td>
<td>not portable, even larger files than for the MS Word format.</td>
<td>same example with graphics.</td>
<td>1700 kB.</td>
</tr>
</tbody>
</table>
IGT-Site, a portable DB driven Web Site

Motivation
Aim
Design
User Interface
Implementation
Outlook
Summary
On the server side Active Server Pages (ASP) together with Visual Basic Scripting Edition is used. On the client side mostly HTML 4 is employed. This guarantees that most browsers which fulfill minimal requirements may view the Web site. Minimal requirements include e.g. the support of frames.

Additionally, on the client side also JavaScript is used. However, it is envisaged to use JavaScript just for enhancements, not for the basic features. This means that if a browser does not support JavaScript he is still able to display all information available on the Web site. However, with support of JavaScript some additional features are activated, as e.g. the automatic updating of the table of contents (see below).

Unfortunately, it seems that Netscape Navigator v.3 does support JavaScript, but not in a correct way. Therefore we exclude support of Netscape Navigator v.3, which seems not to be too restrictive in view that v.4 is already available, and which is, in contrast to v.4, even free.

On the client side we rely also on Cascading Style Sheets, but in view of the different interpretations by the individual browsers we just used a subset, which enables us to use their advantages, without having any drawback due to the not yet released standard.

As a rule all "normal" pages are HTML pages but not ASP pages. The reason is that for multiple accesses of the same page, HTML pages are not reread from the server, but just fetched from the local cache, which improves speed considerably. ASP pages are always reread and only embedded graphics are stored in the local cache.

The disadvantage is that the flexibility of HTML pages is much smaller. It is e.g. quite difficult to avoid error messages popping up when using variables in HTML files under JavaScript1.1.
The first page loaded when attaching the *IGT-Site* (DEFAULT.HTM) serves just to inform the user where he is connected to. The size of this page should be very small to load as quickly as possible. We therefore just display an initial welcome screen (bitmap) which is attractive but small (10 kB).

From there after a few seconds the user is automatically routed to DEFAULT_FRAME.HTM, which divides the screen into two frames:

- the frame with NAME="WWW_Head" fills the upper part. In this frame the page SRC="DEFAULT_HEAD.HTM" is loaded which is static, i.e. the same file is displayed during the whole session.
- the frame with NAME="WWW_MainBody" fills the lower part. In this frame user-choosable HTML files are loaded (e.g. SRC="WWW/HOME.HTM") which divide the frame into two vertical frames
  
  - the frame with NAME="WWW_TOC" fills the lower left part. In this frame the table of contents (TOC) is displayed. By convention, TOC files use the same filename as their parent in WWW_MainBody, but with an additional T as last character (e.g. homeT.htm).
  - the frame with NAME="WWW_Body" fills the lower right part. In this frame the actual content or body is displayed. By convention, body files use the same filename as their parent in WWW_MainBody, but with an additional B as last character (e.g. homeB.htm).

If the frame organization is used as explained above for each file displayed in WWW_MainBody a corresponding help file may be displayed. The name of this help file is assembled from the path of the file in WWW_MainBody (which must be below /WWW/), and /HELP, and the name of the file in WWW_MainBody.
The virgin *IGT-Site*, i.e. without any special pages for e.g. presenting the company, consists approximately of the following code:

- **DB Frontend for administrators:** 1300 lines of code, > 40 forms, 40 defined queries
- **DB Frontend for users:** subset of Frontend for admins with adaptations
- **DB Backend:** 37 tables
- **WWW-Server ASPs:** 7000 lines
- **WWW-Server Style Sheets:** 1000 lines
- **WWW-Server HTML pages:** 7000 lines
In section "Design Concept" it was briefly mentioned, that the database consists logically of two independent parts: the data store itself, called the Backend, and the program where users input their data, the so-called Frontend.
Design of the DB Backend

- **Types of information**
  - Specific types: People, Publications, Projects, Courses, Services, Products
  - General type: Infos

- **Field types**
  - Internal: descriptions, abstracts, keywords, etc.
  - Links: resources (e.g. full papers)

- **Implementation**
  Any relational, SQL compliant database, e.g. SQL-Server or MS-Access

Generally, for all types descriptions, abstracts, keywords for searching, and links to accompanying resources may be specified. These resources (e.g. a full paper) are not stored within the DB itself. From the Frontend the user may specify the local path of a resource he would like to include. The resource is then automatically copied by the Frontend to the *IGT-Site* and a link is placed in the DB.

This design has two main advantages:

- The size of the DB itself remains limited, manipulations of the data (retrieving, etc.) will therefore be much faster.
- The content of the DB is in ASCII. Special search algorithms have been developed for well-defined searches taking into account the logical content of the data.
- Because additional resources are stored within the NT file system third party search engines may be used which support the multitude of past and future file formats in use.

All the types listed above are tailored for a specific purpose, e.g. information about people. All data are input from the Frontend using specially tailored, and thus very easy to handle, forms. Their advantage is simplicity, however reached by limited capabilities. To overcome the latter an additional type was created, called

- INFOS. This type is quite general. Several fields have been foreseen where even the field title may be specified. In this way it is possible for a user to define a multitude of independent data categories. An example of this may be a category of important Web links, or a software database. These categories may then be included under different titles in the table of contents (e.g. “Links” or “Tutorial” in Fig. 4). Further information about INFOS is provided under the heading "Database Field Type: INFOS" below.
Logically the most important tables which correspond to the above-mentioned types are

TPersons
TPublications
TProjects
TCourses,
TServices
TProducts
TInfos.

To avoid redundancy and many-to-many relations, several other tables have been defined, which are linked together in a relational way. An excerpt of some tables and their interrelationships is displayed.
Implementation of the DB

• **Design**
  - Backend: relational data store
  - **Frontend**: forms to access the backend
  - Control of Functionality

• **Security Dispositive**
  - Backend: design protected, data editable by Frontend only
  - Frontend: design protected, data input by forms only
  - Deployment of User Frontend
  - Quality Control of Content
The Frontend relies on MS Access (v.97 or above) and MS Visual Basic for Applications. It is less used for storing data (it actually stores just two variable values: the name of the local user which last used the Frontend, and the database path used for the last connection), but it was chosen due to its powerful support of forms and its tight connection with the data store. In case the need would arise that it should be used from other operating systems than MS Windows, it could be converted to be used with other programs, or even from a Web browser, because it is completely independent of the Backend.
The Frontend is the means for changing contents by "normal" users. A security dispositive must envisage two aims: firstly the Frontend itself must be absolutely protected from any unwanted internal manipulations which could change its design or its local data. And secondly "allowed" changes of the content of the DB should obey the rules and guidelines of the company for its Web site. It is obvious that the latter cannot automatically be guaranteed, but the whole update history may automatically be logged. If a violation of the rules is detected it may at least be traced back to who was responsible for which change.

As an additional safety measure and to inform the Webmaster immediately about all changes, they may automatically be e-mailed to the Webmaster. Programmatically.

**Frontend:**
1. avoid circulation of virgin Frontends
2. MDE versions only
3. Protect each record (on a form base)
4. Byepass Shift-key (-> forms only approach)
5. Disconnect cached DB

**Backend:**
username/password
Implementation

Web Implementation
  Frame Organization
  Table of Contents
  Context-sensitive Help

Database
  Design
  Deployment of the User Frontend
  Security Considerations

Setup of IGT-Site
  Setup of the NT Servers
  Setup of Internet Information Server (IIS)
  Adaptations for the Company
Setup of IGT-Site

- Setup of the NT Servers
  - Domain Server: account "IGT" for Intranet
  - Web Server: account "AnonWWW"
    shares IGT_Site, PersonalResources

- Setup of Internet Information Server (IIS)
  Execute set of (WinBatch-) batch files:
  - copy directory tree and set permissions
  - create account "AnonWWW" and shares
  - setup Site and virtual directories

- Adaptations for the Company
  Company acronym (variable in JavaScript)
  Username/pw for Backend, Frontend, Web access

The **IGT-Site** is hosted on an MS NT Server v.4 (Web Server), which is connected to an NT Domain Server, from which it gets the account information. On the latter a group "IGT" must be defined, to which all company members belong.

On the Web Server the account "AnonWWW" for an anonymous user should be created.

The **IGT-Site** relies on the MS Internet Information Server (IIS) v.4 under MS NT. It may be downloaded for free as part of the MS NT 4.0 Option Pack.

Thanks to its general design the source of the **IGT-Site** may easily be adapted to reflect the specific data of a company. The whole folder structure with all files may be copied to a server where the MS **Internet Information Server** is installed, and after configuring the Web site as outlined above (or with the batch files respectively) the adapted Web site should run smoothly.

Adaptations of the source code concern
1. the acronym of the company running the **IGT-Site** (default: "IGT"),
2. some path information (default: "Website"),
3. the master password for changing personalized User Frontends (default: "master"),
4. the DBPassword of the database (default: "password"), and possibly
5. the "FrontendUsername" and the "BrowserUsername" (default: "admin").
MS Access supports the LIKE search procedure, which searches Umlauts and similar characters in a tolerant way. Unfortunately, this feature is not supported by the current ODBC driver.

For searching people we therefore implemented our own small search engine which skips vocals and characters of classes "c", "n" and "y". Characters at the beginning of a search string are never skipped. This engine usually leads to good results, but sometimes unexpected matches may occur.
Summary of IGT-Site

- **Motivation**: ETH, maintenance, content
- **Aim**: General, content, form
- **Design**: why a DB, connectivity, access
- **User Interface**: Web and DB interface
- **Implementation**: Frames, DB, setup
- **Outlook**: Tolerant search, Frontend, DHTML

The impulse to developing a portable Web site was the immediate need of the Division of Geotechnical Engineering to overhaul its four years old Web site. But as expected from a research organization we did not want to produce just some pages for the public, but we took the opportunity to investigate the various approaches for building a Web site from a more general point of view.

The result is a quite general Web site which is portable and could be used by a variety of organizations. It is tightly connected to a database, where most of the content of the Web site is stored. Inputting data to the Web site may be done also by untrained personnel thanks to an easy-to-handle form based user interface. This has the advantage that the work of updating the content of the Web site is not concentrated in a single person, usually the Webmaster, but may be spread to practically all members of the organization, each of which may be made responsible for that content he is responsible for or interested in also during his professional work. It is expected that due to this concept the content of the Web site should always be up-to-date.

From a technical point of view the Web site is based on Microsoft’s *Active Server Pages*, which run on an NT-Webserver and produce pure HTML code which is sent to the browser. With this approach firstly no additional software is needed on the client side, and secondly basically all browsers running under any operating system are supported.
Thank you for your patience ...