SCIENTIFIC PAPERS
OF THE UNIVERSITY OF PARDUBICE
Series B
The Jan Perner Transport Faculty
11 (2005)

INTEGRATING DATA SOURCES

Karol MATIAŠKO, Martina BABIŠOVÁ, Monika VAJSOVÁ

Department of Informatics, Faculty of Management Science and Informatics, University of Žilina, Slovak Republic

Abstract

Integrating heterogeneous data sources is a big deal for most companies, organizations, universities. Thanks to this every organization can yield the most important and relevant information for every user (in sence of employee, manager, student, professor, ...) without bigger effort.

Keywords**:** *portal, security, personalized view, data representation, XML, RSS, databases, LDAP, Hibernate, AJAX, classic web application model, authentication*

1 Introduction

Nowadays the fundamental objectives of portal solutions are to unify the heterogeneous sources and to represent them in personalized or impersonalized form to user. Data that are represented in such a portal have often different structure, quality, availability and reliability. Sources from which these data comes to home site (place where the portal is localized) are often autonomous sites distributed over LAN and/or not so rare over WAN. These sites have different level of security access, they often have no knowledge about the home site and their main functionality is to offer information in some structure. To select only the relevant, reliable, safe and actual information for the portal's user is a task of the portal's design analysis.

Figure 1.1 shows possible global view on such a system.

The kernel of the portal is composed from moduls, which are assigned to represent and administrate information. Over this kernel are more interfaces that allow the communication with other sites and the representation of the gained information to the user. Not every user can and need to have access to every message on the portal. For these reasons each portal should support personalised access to data. The next important point is security. The same or higher level of security as on the portal should be required from asked site. If it is not possible to achieve this, than we should take data only from trustworthy site.

In the next sections we will write about these parts and interfaces for integrating some interesting kinds of information in the portal. Because portals can be implemented in several programming languages such as PHP, Java, ASP, .NET… the examples will be every time in some other language.

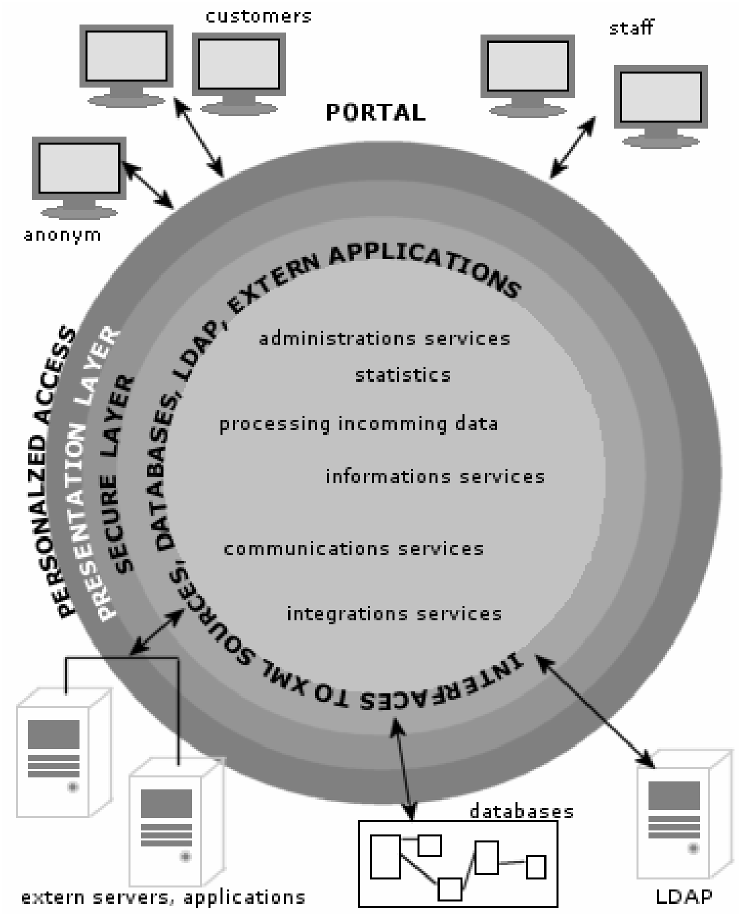

***Fig. 1** Global view on the system*

2 Security

The main quality request on the portal is to secure dataflow from portal to user or application and vice versa. The second request is to display data only to the authorized person.

2.1 Secure dataflow transfer

The first request is well solved through secure connection SSL (Secure Socket Layer). SSL provides endpoint authentication and communications privacy over the Internet using cryptography. SSL encrypts data using public key. By the first connection to client, server sends its public key to client. Information encrypted with public key can be decrypted only with server's private key. Client uses the public key to encrypt its own key and so it isn't possible for the third site to copy client or server. Between client and server enters third site – authentification authority, which assigns the public key with its own signature. In this way no third side can decrypt and read information changed between client and server.

2.2 Safety integrating heterogeneous information data sources

Much information we would like to display on the webportal are located on the site with different security level. These data often flow through non-secure channels.

Between, client sending a request to server and the third side from which should these information come from, should be placed some logic, which should have knowledge about the structure of the requested information and about the client's rights to see this information. Through this logic will be manipulated with data on the server site and processed information will be mediated to the user, but only if he is authorized to see them.

2.3 Authentication

To define, which user already browses over the portal and/or if he is authorized to see some information, is used the authentification. Through login and password every user has to authenticate him-/herself. Then the logic on the server site can define which roles and rights will be assigned to this user. To save and access the user's login and password are often used databases or directories, which are accessed over LDAP. For holding the information about the user during his browsing are mostly used sessions. The example below shows, how could be session implemented in ASP.

Example 2.1
```
<% Session.Timeout=20
   Session("username")="Peter Small"
   Session("group")="teacher"
   Session("departure")="DI" %>
```

For security reasons it is important to set the time period for which the session is valid. If a user has not requested or refreshed a page in the application during this specified period, session will expire and the user will be not for this application authenticated anymore.

How to restrict access to some information for already logged person shows the following example.

Example 2.2
```
<% If Session("group")="teacher" Then %>
  Information only for teacher
<% Else If Session("group")="student" Then %>
  Information only for student
<%      End If
    End If %>
```

For the authenticated user from the Example 2.1 will be the output from Example 2.2:
```
Information only for teacher
```

3 Integration data from LDAP

In Section 2.3 is mentioned than one of the most popular access protocols for data is LDAP.

Lightweight Directory Access Protocol (LDAP) [2] is the core communication and access protocol behind all major directory server products today, including Active Directory.

LDAP provides read as well as write access to a directory, but it was optimized for reading rather than writing.

The organizations structures are very often saved in directory structure and accessed over LDAP. From this structure we can get personal information about the user, then information about departure or structure of departures to which user belongs. In “portal’s language” this can mean groups and subgroups of users. Into personal data belong login and password. These two attributes are most important for authentication. Example 3.1 shows how the interface to LDAP works in PHP.

Example 3.1
```
$host = "ldaps://genesis.fri.utc.sk";
if($ds = ldap_connect($host)) { // connecting to the LDAP
  $base[]="ou=People,dc=fri,dc=utc,dc=sk";
  for ($i = 0; $i < count($base); $i++) {
    $ldap_dn=$base[$i];
    //searching the user
    $r=@ldap_search($ds, $ldap_dn,'uid='.$login);
    if($r) {
      $result = @ldap_get_entries($ds,$r);
      if($result[0]) {
        $ldap_dn=$result[0]['dn'];
        break;
      }
    }
    //binding to the column with password and comparing this with
    //the one from user
    if($r = @ldap_bind($ds, $ldap_dn, $password)) {
      //checking the group, where the user is falling into
      $filter="(uid=$login)";
      $what=array("gidnumber");
      if($r=@ldap_search($ds,$ldap_dn,$filter,$what)) {
        $entries = @ldap_get_entries($ds,$r);
        if ($entries [0]["gidnumber"][0]== 14004) { // is teacher
          $group = "teacher";
        }
        else {
          $group = "student";
        }
      }
    }
  }
```

In this example we've got user group which includes users' personal data which can be saved in session for next usage.

On the similar principle we can show for example all employees from departure "XY", or all students. The problem rise, when we want to list e.g. all students, or all teachers at the university, this takes a long time. The solution can be very easy: The information about students or teachers are not changed very often, so it is possible to script some batch, that will create a file with all students, or all teachers, or all teachers from departure "XY". This batch can be start as often as we need - e.g. every day in the night. With this approach we will not require the list of students directly through LDAP, but we can display the content of the created file.

4 Integration data from databases

Databases are another place, where data about users, portal's structure, users and administrators rights, and the entire portal's content can be saved.

For the safety reasons it is sometimes better to access only the read-only copy of database or only some fragments, views… The read-only copy can be synchronously or asynchronously updated – depending on the content. For example the fruitfulness of the study is counted only one time per academic year. So you need to change this fragment only once a year.

We can access the data from the database either directly through inbuilt functions or through JDBC in case of Java or through ODBC in other cases. Following example show us the manipulation with data through JDBC in Java.

Example 4.1

```
public class UserList {
 public static void main(String[] args) {
  Connection conn=null;
  Statement stat=null;
  PreparedStatement pstat=null;
  try {
   Class.forName("org.firebirdsql.jdbc.FBDriver");
   conn=DriverManager.getConnection(
       "jdbc:firebirdsql:localhost:c:/test.fdb","db","password");
   stat=conn.createStatement();
   ResultSet rs=stat.executeQuery("select * from user");
   showResult(…);
   conn.close();
  } catch (Exception e) {
      …
  } finally{
      …
  }
 }
}
```

For Java environment were developed some interesting possibilities how the information from database can be directly mapped into the objects, which are manipulated in the middleware or directly in presentation layer. Some examples are mentioned below.

HIBERNATE [4] is a powerful, ultra-high performance object/relational persistence and query service for Java. Hibernate lets you develop persistent classes following common Java idiom - including association, inheritance, polymorphism, composition, and the Java collections framework. Hibernate allows you to express queries in its own portable SQL extension (HQL), as well as in native SQL, or with Java-based criteria and example objects. Examples 4.2 and 4.3 show us sample functioning of Hibernate. Mapping muster for database and relations are configured in hibernate.cfg.xml.

Example 4.2

```
<hibernate-configuration><session-factory>
```

```
 <property name="connection.username">root</property>
 <property name="connection.url">
  jdbc:mysql://localhost/clovek
 </property>
 <property name="dialect">
  net.sf.hibernate.dialect.MySQLDialect
 </property>
 <property name="connection.password">password</property>
 <property name="connection.driver_class">
  com.mysql.jdbc.Driver
 </property>
 <!-- mapping files -->
 <mapping resource="sk/example/uni/hibexample/model/Clovek.hbm.xml"/>
</session-factory></hibernate-configuration>
```

File Clovek.hbm.xml contains direct mapping from one database-table “hclovek” to Clovek.java class. Every column is mapped into one attribute of the class.

Example 4.3

```
<hibernate-mapping>
 <class name="sk.siemens.skola.hibexample.model.Clovek"
        table="hclovek"
        dynamic-update="false" dynamic-insert="false">
  <id name="id" column="id" type="long" length="32">
   <generator class="native"></generator>
  </id>
  <property name="meno" type="java.lang.String" update="true"
            insert="true" column="meno"/>
  <property name="narod" type="java.util.Date" update="true"
            insert="true" column="narod"/>
  <many-to-one name="otec"
               class="sk.siemens.skola.hibexample.model.Clovek"
               cascade="none" outer-join="auto"
              update="true" insert="true" column="otec"/>
    .
    .
    .
 </class>
</hibernate-mapping>
```

Through these xml-files we can generate classes. There are some products on the market, which make all the work with configuring the xml-files instead of us.

TORQUE [5] is the next possibility, which was mentioned upper. It is a persistence layer, which generates all the database resources required in our application and includes a runtime environment to run the generated classes. Torque was developed as part of the Jakarta Turbine Framework. It is now separated and can be used by itself. It has also migrated from Apache Jakarta to Apache DB.

On the market there are many products, plug-ins and utilities for manipulating with databases through web interfaces. It depends only on analyses, what everything we need to perform.

5 Integrating XML and RSS data in portal

The next part concerns XML and RSS and their integrating into portal.

Extensible Markup Language (XML) [1] is a data storage toolkit, a configurable vehicle for any kind of information, an evolving and open standard embraced by everyone from bankers to webmasters. With its clear, simple syntax and unambiguous structure, XML is easy to read and parse by humans and programs alike.

Rich Site Summary (or Really Simple Syndication) is a dialect of XML (its in format which responds to XML 1.0 specification) and serves to allocation of the content of coverage's and information's portals.

The following Example 5.1 shows integrating of XML sources into intranet portal.

Example 5.1

```
<?xml version="1.0" encoding="windows-1250"?>
 <nbsDailyFxRateList>
  <lang>slovenčina</lang>
  <validFrom>YYYY-MM-DD</validFrom>
  <number>LLL</number>
  <rateList>
   <rate>
    <country>BBBBBBBBBBBBBBB</country>
    <ccyCode>EEE</ccyCode>
    <amount>KKKK</amount>
    <value>HHH.HHH</value>
   </rate>
      ▪▪
   </rateList>
  </nbsDailyFxRateList>
```

Every programming language has its own integrated functions, which implement XML parser and some functions for manipulation with the content. For this example we show us the solution of integrating XML source into portal for PHP language.

Example 5.2
```
<?php
$xml_parser = xml_parser_create();
xml_parser_set_option($xml_parser,XML_OPTION_CASE_FOLDING, true);
xml_set_element_handler($xml_parser, "startElement", "endElement");
xml_set_character_data_handler($xml_parser, "characterData");

//implementation of the function startElement
function startElement($parser, $name, $attrs)
{
  global $currentTag;
  $currentTag = $name;
}
//implementation of the function endElement
function endElement($parser, $name)
{
  global $courses, $country, $ccyCode, $amount, $value;
  if(strcmp($name, "RATE") == 0)
  {
    $courses[] = array{"country" => $country,
                       "ccyCode" => $ccyCode,
                       "amount"  => $amount,
                       "value"   => $value};
  }
}
//implementation of the function characterData
function characterData($parser, $name)
{
  global $country, $ccyCode, $amount, $value, $currentTag;
  if(strcmp($currentTag, "COUNTRY") == 0)
  {
    $country .= $data;
  }
  if(strcmp($currentTag, "CCYCODE") == 0)
  {
    $ccyCode .= $data;
  }
  if(strcmp($currentTag, "AMOUNT") == 0)
  {
    $amount .= $data;
  }
  if(strcmp($currentTag, "VALUE") == 0)
  {
    $value .= $data;
  }
}
?>
```

After this, it follows the manipulation with the array $courses.

Processing of the files in RSS format is very similar to the manipulation with XML files. RSS has depending on the version fixed structure. The example of RSS file:

Example 5.3
```
<rss version="0.92">
  <channel>
    <title>Java Technology Headlines</title>
    <link>http://developers.sun.com/rss/java.xml</link>
    <description>Technical content and news from java.sun.com, the
     premier source of information about the Java platform.
    </description>
    <language>en-us</language>
    <image>
      <title>java.sun.com</title>
      <url>http://developers.sun.com/im/logo_java_grey.gif</url>
      <link />
      <width>144</width>
      <hight>40</hight>
      <description>Visit java.sun.com</description>
    </image>
    <item>
      <title>The New Modality API in Mustang</title>
      <link>
        http://java.sun.com/developer/technicalArticles/J2SE/Desktop/
        mustang/modality/?feed=JSC
      </link>
      <description>
         Learn how Java SE 6 (code name Mustang) allows greater
         functionality for dialog boxes, supporting modeless, document,
         application, and toolkit modality to ease the user's
         experience.
       </description>
       <date />
     </item>
       ▪
   </channel>
</rss>
```

How we can see, the result of manipulating with RSS is a list of url-links with short description. All the content of these links is saved on remote site.

6 Integrating application in portal

In some cases before the portal will be created, there are some applications, modules, which are already in usage of the organization and do not need to be changed. Or in other case, there can be some applications that are managed from some extern company. In such a case these applications need to be only integrated and not changed. For these reasons we only need to share authentication's data about user with these applications.

For example we would like to use web-mail in the portal as one module and our request is, that person which is already authenticate in the portal, will be authenticated in the web-mail module too. Internet community provides plenty of different web-mail modules, which can be chosen. To use some of them for the portal with this requirement, we only need "to pass" authentication information to the module. This is well done through session.

7 Classic web application model versus AJAX

In term of integrating heterogeneous information sources into the web application, it makes sense to point on AJAX [6] [7] [8]. AJAX (Asynchronous JavaScript and XML) represents a generic application model that would enable more interactive, more responsive, and smarter web applications.

Some uses for AJAX interactions are the following:

1. Real-Time Form Data Validation: Form data such as user IDs, serial numbers, postal codes, or even special coupon codes that require server-side validation can be validated in a form before the user submits a form.

2. Auto completion: A specific portion of form data such as an email address, name, or city name may be auto completed as the user types.

3. Master Details Operations: Based on a client event, an HTML page can fetch more detailed information on data such as a product listing that enables the client to view the individual product information without refreshing the page.

4. Sophisticated User Interface Controls: Controls such as tree controls, menus, and progress bars may be provided that do not require page refreshes.

5. Refreshing Data on the Page: HTML pages may poll data from a server for up-to-date data such as scores, stock quotes, weather, logged users or application-specific data.

6. Server-side Notifications: An HTML page may simulate a server-side push by polling the server for event notifications that may notify the client with a message, refresh page data, or redirect the client to another page.

The classic web application model adopts a "click, wait, and refresh" user interaction paradigm and a “synchronous request/response” communication mechanism (Figure 2). "Click, wait, and refresh" and "synchronous request/response" result in slow, unreliable, low productivity and inefficient web applications. These two basic behaviors must be altered to produce higher performance, more interactive, more efficient web applications - precisely what the AJAX application model (Figure 3) does. In the AJAX model:

1. "Partial screen update" replaces the "click, wait, and refresh" user interaction model. During user interaction within an AJAX-based application, only user interface elements that contain new information are updated; the rest of the user interface remains displayed without interruption. This "partial screen update" interaction model not only enables continuous operation context, but also makes non-linear workflow possible.

2. Asynchronous communication replaces "synchronous request/response model." For an AJAX-based application, the request/response can be asynchronous, decoupling user interaction from server interaction. As a result, the user can continue to use the application while the client program requests information from the server in the background. When new information arrives, only the related user interface portion is updated.

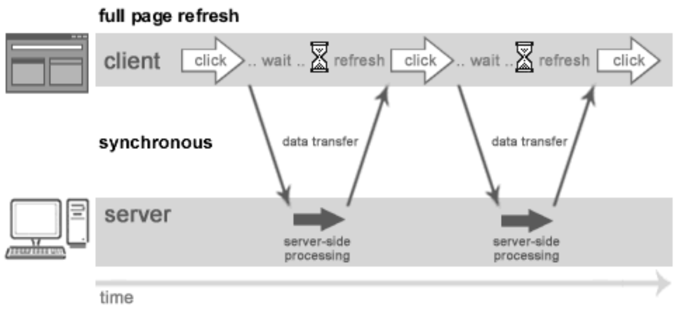

Fig. 2 *Classic web application model: Full page refresh and Synchronous Communication*

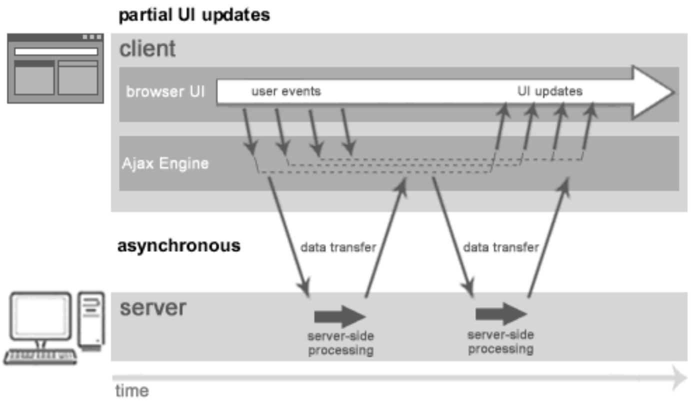

Fig. 3 *Ajax model: Partial UI Updates and Asynchronous Communications*

One of the keys to the AJAX approach is the XMLHttpRequest object in JavaScript. The XMLHTTPRequest object enables JavaScript to make HTTP requests to a remote server without the need to reload the page. In essence, HTTP requests can be made and responses received, completely in the background and without the user experiencing any visual interruptions.

With the following example, we show us how it works. In this example it will be verified the validity of user ID. To the text field for user ID is dedicated event “onkeyup”:

```
<input type="text" id="userid" name="id" onkeyup="validate();">
```

The `validate()` function creates an `XMLHttpRequest` object:

```
var req;
function validate() {
   var idField = document.getElementById("userid");
   var url = "validate?id=" + escape(idField.value);
   if (window.XMLHttpRequest) {
       req = new XMLHttpRequest();
   } else if (window.ActiveXObject) {
       req = new ActiveXObject("Microsoft.XMLHTTP");
   }
   //HTTP method, url and a boolean indicating whether or not the call
   //will be made asynchronously
   req.open("GET", url, true);
   //If an interaction is set as asynchronous (true) a callback
   //function must be specified.
   req.onreadystatechange = callback;  req.send(null);
}
```

A servlet mapped to the URI "validate" checks whether the user ID is in the user database.

```
public class ValidateServlet extends HttpServlet {
  private ServletContext context;
  public void init(ServletConfig config) throws ServletException {
    this.context = config.getServletContext();
  }
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
    throws IOException, ServletException {
    String targetId = request.getParameter("id");
    //The developer must be aware of two things: the Content-Type
    //must be set to text/xml. Second, the Cache-Control must be set
    //to no-cache.
    response.setContentType("text/xml");
    response.setHeader("Cache-Control", "no-cache");
    if(targetId != null)
    {
       //User ID can be checked from database or from LDAP…
       Class.forName("org.firebirdsql.jdbc.FBDriver");
       conn=DriverManager.getConnection(
            "jdbc:firebirdsql:localhost:c:/test.fdb","db","password");
       stat=conn.createStatement();
       ResultSet rs=stat.executeQuery(
            "select * from user where userid = targetId");
       if ((rs.next())){
         response.getWriter().write("valid");
       } else {
         response.getWriter().write("invalid");
       }
    } else {
      response.getWriter().write("invalid");
    }
  }
}
```

The `XMLHttpRequest` object was configured to call the `callback()` function when there are changes to the `readyState` of the `XMLHttpRequest` object.

```
function callback() {
 //readyState is 4 -> signifying the XMLHttpRequest call is complete
 if (req.readyState == 4){
  //The HTTP status code of 200 signifies a successful HTTP interaction
  if (req.status == 200) {
   //update the HTML DOM based on whether or not message is valid
  }
 }
}
```

Following XML document is returned from the `ValidateServlet`:

```
<message>valid</message>

function parseMessage() {
  var message = req.responseXML.getElementsByTagName("message")[0];
  setMessage(message.childNodes[0].nodeValue);
}
```

The `parseMessages()` function will process an XML document retrieved from the `ValidateServlet`. This function will call the `setMessage()` with the value of the message element to update the HTML DOM. This message can be displayed statically, altering existing HTML element or dynamically, creating new element. In our example we show us statically solution with existing `<div>` element.

```
<div id="userIdMessage"></div>

function setMessage(message) {
  mdiv = document.getElementById("userIdMessage");
  if (message == "invalid") {
    mdiv.innerHTML = "<div style=\"color:red\">Invalid User ID</ div>";
  } else {
    mdiv.innerHTML = "<div style=\"color:green\">Valid User ID</ div>";
  }
}
```

The weakness of using AJAX is the support for JavaScript technology DOM APIs which can differ in various browsers, so we must take care when developing applications.

8 Conclusion

The goal of this paper was to show, what everything can be integrated in portal and how easily it can be done. The biggest problem for us is to choose the most effective and functional technique for solving our problems.

Lektoroval: Ing. Jiří Netolický

Předloženo: 7.7.2006

References

1. RAY, E. *Learning XML, First Edition*, January 2001
2. ALLEN, R., PUCKETT, R. *Managing Enterprise Active Directory With Lightweight Directory Access Protocol (LDAP),* Addison Wesley Professional, April 2002
3. POŠMURA, V. *Apache Príručka správce WWW serveru*, Computer Press, Praha, 2002
4. http://hibernate.bluemars.net/
5. http://db.apache.org/torque/
6. http://www.developer.com/design/article.php/10925_3526681_1
7. http://java.sun.com/developer/technicalArticles/J2EE/AJAX/
8. http://www.janinedalton.com/blog/archives/2005/ajax-asynchronous-javascript-and-xml/
9. http://portals.apache.irg

Resumé

INTEGRÁCIA DÁTOVÝCH ZDROJOV

Karol MATIAŠKO, Martina BABIŠOVÁ, Monika VAJSOVÁ

Príspevok dokumentuje možnosti integrácie dátových zdrojov pri tvorbe rozsiahlych informačných systémov, ktoré čerpajú dáta z rôznych a popritom heterogénnych zdrojov. Článok demonštruje vybrané spôsoby integrácie tak, aby mohli byť prezentované v portálových riešeniach.

Summary

INTEGRATING DATA SOURCES

Karol MATIAŠKO, Martina BABIŠOVÁ, Monika VAJSOVÁ

Integrating heterogeneous data sources is a big deal for most companies, organizations, universities ... Thanks to this every organization can yield the most important and relevant information for every user (in sense of employee, manager, student, professor,...) without bigger effort. Our paper describes what everything can be integrated in portal and how easily it can be done. The biggest problem for us is to choose the most effective and functional technique for solving our problems.

Zusammenfassung

DIE INTEGRATION VON HETEROGENEN DATENQUELLEN

Karol MATIAŠKO, Martina BABIŠOVÁ, Monika VAJSOVÁ

Der Beitrag darstellt die Integrationsmöglichkeiten von den versiedenen unabhängigen und nebenbei heterogenen Datenquellen bei den Entwurf der Informationsystemen. Artikel demonstriert ausgewählte Verfahrens, die ermöglichen die heterogene Quellen einfach an einem Webportal zusammenzufassen.