Developing a Mobile Web-based Application with Oracle9*i* Lite Web-to-Go

Christian Antognini
Trivadis AG
Zürich, Switzerland

Introduction

More and more companies need to provide their employees with full access to their enterprise databases when they are not directly connected to the company's network. In many cases a virtual private network over a dial-up or wireless connection can solve such a problem. Unfortunately this method doesn't provide a solution when:

- the needed network bandwidth is not available or is too expensive,
- neither a dial-up nor a wireless connection can be guaranteed at anytime and everywhere.

In addition, sometimes the data is mostly read-only and therefore a remote connection is not an effective or efficient way to access it.

Therefore the need for an application which can be used when the users are disconnected from the company's network is a real one. Of course such applications lead to another set of problems that must be solved! The biggest problems are the following.

- Since the users are working offline, the data must be replicated. This replication of read-only data is not really a problem, but when the data must also be modified by the clients the complexity increases.
- The users should be able to use the same application while they are working online and offline. Notice that is not a technical problem to develop such an application, but the overhead for the developers can be huge (i.e. high cost of development).
- The deploying and management of the application must be centralized.

To effectively face up to these problems, Oracle recommends Oracle9*i* Lite. In this article, we will discuss the main features of this product and show a possible architecture based on one of its components.

Oracle9*i* Lite

Oracle9*i* Lite is not only a small footprint database system, but is a complete framework for creating mobile applications based on Oracle databases. It can be used to develop fat clients or thin clients (i.e. web-based applications). As emphasized by the title of this article, here we are interested in the latter. Therefore we will only cover the features of Web-to-Go, i.e. the component of Oracle9*i* Lite that enables this type of application development.

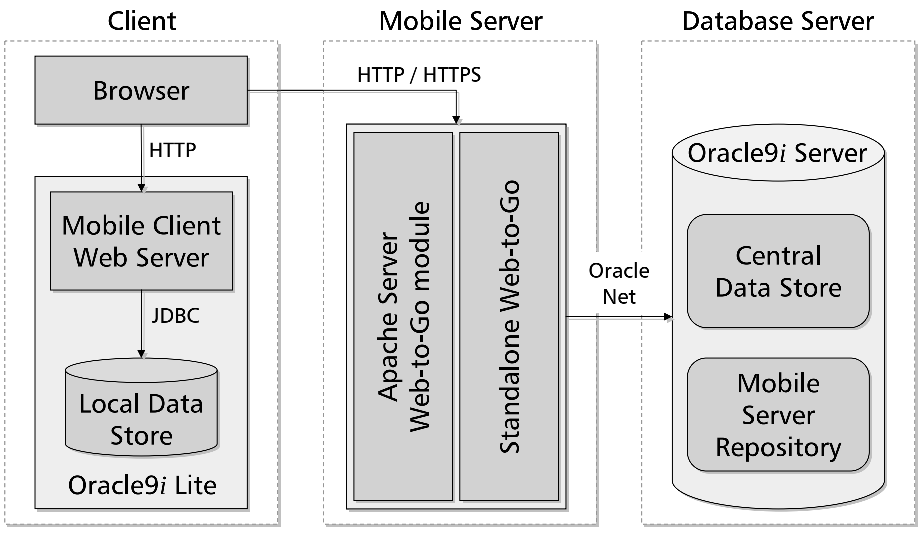
What is Web-to-Go?

Web-to-Go is a framework that enables developers to create web-based applications that can be used either online (i.e. while connected to the company's network or the Internet) or offline. Therefore the main features of Web-to-Go are the following:

- the replication and synchronization of data used by offline clients,
- the development of a single application that can be run either online or offline,
- a central management of the application (i.e. the management of the software itself, the data and the users).

The Web-to-Go applications are based on standard J2EE components like Servlets and JSPs. Presently the product is available for Windows 32, Windows CE/Pocket PC, Palm OS and EPOC. However this article only covers the Windows 32 version.

Web-to-Go Three-tier Environment



Client	When the user works online, only the web browser is used, i.e. each call is sent to the Mobile Server. When the user works offline, the Mobile Client Web Server is used to run the application's components that access the local database. For each user working on a specific client a local database is created.
Mobile Server	This runs the application's components used by the online users and provides seamless synchronization between the clients' Oracle Lite databases and the Oracle Server. It can be integrated with an Apache server via module or it can be started standalone.
Database Server	The Mobile Server repository and the application data must reside in the same Oracle database. The Mobile Server repository contains the properties of the application and the application's components themselves.

Note: Oracle8, Oracle8*i* and Oracle9*i* are supported on the database server (notice that Oracle9*i* is supported since Oracle9*i* Lite 5.0.1 only).

Lite Database

The Lite database is used to store the data on the client when the user is working offline. Of course the schema installed within this database should be compatible with the schema installed in the company's database server. Therefore it supports the main objects that can be used with the Oracle Server database, i.e.:

- Tables with constraints, indexes and triggers
- Views
- Sequences
- Synonyms
- Java stored functions and procedures
- Users

Notice that PL/SQL stored functions, procedures and packages are not supported!

Of course different tools are provided to manage the database. The most important are the following:

• CREATEDB	used to create the database
• MSQL	command-line interface to connect a database
• OLLOAD	command-line tool to load data from an external file into a table, or to unload (dump) data from a table to an external file

Note that Oracle Net (formerly Net8 and SQL*Net) cannot be used to connect the database. Instead an ODBC interface is provided.

Building Web-to-Go Applications

Web-to-Go applications adhere to web standards. Therefore they are composed of static objects like HTML/GIF/JPEG files and dynamic components like Servlets, JSPs, JavaBeans and Applets.

The application's components can be developed and debugged with any Java IDE.

To use Web-to-Go features or to simplify development (for example to generate HTML code from database tables), some Java packages are provided. In any case, the developer is not compelled to use these packages; although it can develop standard Java components or use other libraries. Getting the connection to the database is an exception to this. In fact the applications must transparently use both databases, therefore it is not possible to carry out a standard connection. The following code shows how to get a connection in a Servlet:

```
import oracle.lite.web.servlet.OraHttpServletRequest;
import java.sql.Connection;

public class MyServlet extends HttpServlet
{
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
                    throws ServletException, IOException
  {
    ...
    Connection connection = ((OraHttpServletRequest) request).
                            getUserProfile().getConnection();
    ...
  }
}
```

Notice that the connection itself is a standard JDBC connection. Therefore these are the only lines of codes that are specific to Web-to-Go.

Deployment

The deployment must be performed with the tool WTGPACK (of course distributed with the development kit). This tool is basically used to specify which components are part of the application and which objects must be replicated between the databases.

During the development the Mobile Server Repository is not used. Instead WTGPACK generates an XML configuration file. Also for this reason, a "development" standalone Web-to-Go server, that reads the XML configuration file, is available for development.

Of course at any moment it is possible to deploy the components in the Mobile Server Repository and test them via the standard Web-to-Go server.

Client Installation

To perform the installation of the Mobile Client for Web-to-Go, a small (113KB) setup program must be downloaded from the Mobile Server. This program automatically downloads the necessary software and performs its installation. Notice that this installation is very compact. In fact less then 5MB are necessary for the Mobile Client Web server and the "database engine".

Then the first time a user connects from a specific mobile client, the application is downloaded and the data is synchronized.

Data Synchronization

The most important thing to notice, is that the standard Oracle replication is not used here! Instead Web-to-Go uses a model similar to publish-subscribe.

During data synchronization, the mobile client uploads data changes into IN queues in the Oracle Server database. On the database server, a background process (the consolidator message generator and processor - MGP) applies the pending transactions from the IN queues to the Oracle database. The MGP is also responsible for generating new data updates for the mobile clients. These updates are stored in OUT queues from where the mobile client retrieves them during the synchronization. Notice that it is possible to upload changes into the IN queue and download updates from OUT queue even if the MGP is not running.

The network load generated during data synchronization is modest. For example an update of 1000 rows in a table on the server (ca. 100KB of raw data) has generated 120KB of traffic between the client and the server. Therefore the synchronization of "small" amounts of data should work without problems also when the client is connected to the Internet via a 56K modem.

Unfortunately only two conflict resolution methods are available, i.e. client wins and server wins. Therefore for specific conflict resolutions, triggers on the server must be implemented.

Management

Management is carried out via the Mobile Server Control Center, a Web interface that enables the administrator to perform different operations like:

- administering the Mobile Server itself (starting/stopping MGP, view active sessions, etc.),
- managing the users and their privileges,
- managing the applications (modifying application properties, suspending/resuming an application, etc.),
- scheduling data synchronization jobs for specific mobile clients.

Conclusion

Considering the fact that standard J2EE components are used, the ease of Web-to-Go's management and client installation, the capacity of its deployment, the quality of its data synchronization and its high overall performance, this product is certainly worth considering if you have to develop of a mobile web-based application.