Ecole Supérieure privée d'ingénierie et de technologies

Mini Java Project

HyperPath GPS based service discovery

Authors:
Adel ATTIA
Chedi TOUEITI

Supervisor:
Mr :
Mohamed Ali BETTAIEB

June 6, 2011

Abstract

Motivation: Why do we care about the problem and the results? If the problem isn't obviously "interesting" it might be better to put motivation first; but if your work is incremental progress on a problem that is widely recognized as important, then it is probably better to put the problem statement first to indicate which piece of the larger problem you are breaking off to work on. This section should include the importance of your work, the difficulty of the area, and the impact it might have if successful.

Problem statement: What problem are you trying to solve? What is the scope of your work (a generalized approach, or for a specific situation)? Be careful not to use too much jargon. In some cases it is appropriate to put the problem statement before the motivation, but usually this only works if most readers already understand why the problem is important.

Approach: How did you go about solving or making progress on the problem? Did you use simulation, analytic models, prototype construction, or analysis of field data for an actual product? What was the extent of your work (did you look at one application program or a hundred programs in twenty different programming languages?) What important variables did you control, ignore, or measure?

Results: What's the answer? Specifically, most good computer architecture papers conclude that something is so many percent faster, cheaper, smaller, or otherwise better than something else. Put the result there, in numbers. Avoid vague, hand-waving results such as "very", "small", or "significant." If you must be vague, you are only given license to do so when you can talk about orders-of-magnitude improvement. There is a tension here in that you should not provide numbers that can be easily misinterpreted, but on the other hand you don't have room for all the caveats.

Conclusions: What are the implications of your answer? Is it going to change the world (unlikely), be a significant "win", be a nice hack, or simply serve as a road sign indicating that this path is a waste of time (all of the previous results are useful). Are your results general, potentially generalizable, or specific to a particular case?

Thanks

From Adel

Attia Adel

To all the Open Source projects and communities that make knowledge accessible to everyone and our lives as developers more challenging and fulfilling.

Chedi Toueiti

Contents

List of Figures

Chapter 1

Introduction

1.1 Project purpose

1.1.1 Project definition

This project aims to address the work-flow surrounding the acquisition, analysis and reporting of GPS data collected via a mobile device and exposing these data to the final user using a custom back office interface.

The finality of the process is to provide a complete solution for monitoring mobile target and acquiring instant knowledge of their current position and past movements history.

1.1.2 Challenges

In the current days geo-localization have became a key feature for many industries, as it provide ways to track targets and by extension collect valuable data about their behaviors. Such data are used for logistics management and optimization and large amounts are spent every year in improving such technologies and investing in new one (such as the Galileo Stellite system).

Location-based service

A location-based service (LBS) is an information or entertainment service, accessible with mobile devices through the mobile network and utilizing the ability to make use of the geographical position of the mobile device.

LBS can be used in a variety of contexts, such as health, indoor object search, entertainment, work, personal life, etc...[5]

LBS include services to identify a location of a person or object, such as discovering the nearest banking cash machine or the whereabouts of a friend or employee. LBS include parcel tracking and vehicle tracking services. LBS can include mobile commerce when taking the form of coupons or advertising directed at customers based on their current location. They include personalized weather services and even location-based games. They are an example of telecommunication convergence.[6]

In the recent past, price (especially the hardware cost) have been the main constraint for many small companies and individuals and the field had not retained the due attention and publicity. But, with the tsunami of mobile devices and smart phones equipped with GPS sensor and network connectivity on the market, the

geo-localistion is back and the applications are more proliferous than ever. That's where the first challenge reside:

- using a mobile device, acquire GPS data and send them to the main server for storage and analysis.
- store the GPS data in the device in case of missing network connectivity and send them back to the server when restored.
- receive GPS data from the server and use them to guide the target through a specific path

Because data without interpretation is merely obscure sequence of numbers, any decent geo-localization system must have a complete set of tool for the reception, storage, analysis and custom business logic. That's why the proposed system will be provided with back office for managing the following tasks:

- Receive and save the GPS data send by the mobile target in a persisting storage system (database, file...)
- Using third party map data, render a graphical representation of the current target position and the past movements history of one or more target.
- Based on the user data, compute paths and send them as GPS coordinate to the mobile target.
- Provide complete reporting schema using the data acquired via the mobile target.

1.2 State of the art

1.3 Solution

1.4 Introduction to Android

1.4.1 History of Android

Historically, developers, generally coding in low-level C or C++, have needed to understand the specific hardware they were coding for, generally a single device or possibly a range of devices from a single manufacturer. As hardware technology and mobile Internet access has advanced, this closed approach has become outmoded.

In more recent years, the biggest advance in mobile phone development was the introduction of Java hosted MIDlets. MIDlets are executed on a Java virtual machine, a process that abstracts the underlying hardware and lets developers create applications that run on the wide variety of devices that supports the Java run time. Unfortunately, this convenience comes at the price of restricted access to the device hardware.

Google acquired the start-up company Android Inc. in 2005 to start the development of the Android Platform. The key players at Android Inc. The Android SDK was first issued as an *"early look"* release in November 2007. In September 2008, T-Mobile announced the availability of the T-Mobile G1, the first smartphone based on the Android Platform. A few days after that, Google announced the availability of Android SDK Release Candidate 1.0. In October 2008, Google made the source code of the Android Platform available under Apache's open source license.

Android sits alongside a new wave of mobile operating systems designed for increasingly powerful mobile hardware. Windows Mobile, the Apple iPhone, and the Palm Pre now provide a richer, simplified development environment for mobile applications. However, unlike Android, they're built on proprietary operating systems that in some cases prioritize native applications over those created by third parties, restrict communication among applications and native phone data, and restrict or control the distribution of third-party apps to their platforms.

1.4.2 Why Android?

Android has the potential for removing the barriers to success in the development and sale of a new generation of mobile phone application software. Just as the the standardized PC and Macintosh platforms created markets for desktop and server software.

Here are several Android platform development advantages you can get if you choose Android:

- **Affordability and customization.** The main Android platform development advantage of the Google Android OS it is open source, meaning that its source code is freely available to developers. And because the code is free, there are no licensing fees to deal with and Android app developers have more leeway in terms of developing apps.

- **A low barrier to entry in the mobile application market.** Because the technology is open sourced, Google Android apps are less expensive to produce compared to other mobile operating systems like the IPhone OS and Windows Mobile. Quite often the primary costs for building apps for Android include development and testing expertise of developers, cost of test devices and royalty fees in case you want to distribute your app to third-party app stores like the Android Market.

1.5 Introduction to Eclipse RCP

1.5.1 Eclipse framework

The Eclipse philosophy is simple and has been critical to its success. The Eclipse Platform was designed from the ground up as an integration framework for development tools. Eclipse also enables developers to easily extend products built on it with the latest object-oriented technologies.

Although Eclipse was designed to serve as an open development platform, it is architected so that its components can be used to build just about any client application. The minimal set of modules needed to build a rich client is collectively known as the Rich Client Platform (RCP).

1.5.2 Why RCP ?

The Standard Widget Toolkit (SWT) is the graphical widget toolkit used by eclipse. Originally developed by IBM, it was created to overcome the limitation if the Swing graphical user interface (GUI) toolkit introduced by Sun. Swing is 100% Java and employs a lowest common denominator to draw its components by using Java 2D to call low level operating system primitives. SWT, on the other hand, implements a common widget layer with fast native access to multiple platforms.

SWT's goal is to provide a common API,but avoid the lowest common denominator problem typical of the other portable GUI toolkits. SWT was designed for the following:

Performance : SWT claims higher performance and responsiveness, and lower system resource usage than Swing.

Native look and feel : Because SWT is a wrapper around native window systems such as GTK+ and Motif, SWT widgets have the exact same look and feel as native ones. This is in contrast to the Swing toolkit, where widgets are close copies of native ones. This is clearly evident just by looking at Swing application.

Extensibility : Critics of SWT may claim that the use of native code does not allow for easy inheritance and hurts extensibility. However, both Swing and SWT support for writing new widgets using Java code only.

1.6 Introduction to GlassFish

1.6.1 Whats a java application server?

A web application is a dynamic extension of a web or application server. Web applications are of the following types:

Presentation-oriented: A presentation-oriented web application generates interactive web pages containing various types of markup language (HTML, XHTML, XML, and so on) and dynamic content in response to requests. Development of presentation-oriented web applications is covered in Chapter 4, JavaServer Faces Technology through Chapter 9, Developing with JavaServer Faces Technology.

Service-oriented: A service-oriented web application implements the endpoint of a web service. Presentation-oriented applications are often clients of service-oriented web applications.

1.6.2 Why GlashFish

GlassFish advantages

With so many options in Java EE application servers, why choose GlassFish? Besides the obvious advantage of GlassFish being available free of charge, it offers the following benefits:

Java EE reference implementation: GlassFish is the Java EE reference implementation. This means that other application servers may use GlassFish to make sure their product complies with the specification. GlassFish could theoretically be used to debug other application servers.

Supports latest versions of the Java EE specification: As GlassFish is the reference Java EE specification, it tends to implement the latest specifications before any other application server in the market. As a matter of fact, at the time of writing, GlassFish is the only Java EE application server in the market that supports the complete Java EE 6 specification.

1.7 Introduction to BIRT

1.7.1 Introduction to business intelligence

To give it a formal definition I would say that business intelligence is any tool or method that allows developers to take data or information, process it, manipulate it, and associate it with related information and present it to decision makers. As for a simplified definition, it's presenting information to decision makers in a way that helps them make informed decisions.

1.7.2 The Current State of the BI Market

you can divide the major players in this field into two categories: commercial offerings and open-source offerings. Each category has its own benefits and drawbacks. With the commercial offerings, typically you have familiar names such as Actuate and Business Objects,offering various tools aimed at different levels of business. Some of these tools are large and enterprise reporting platforms that have the ability to process,analyze, and reformat large quantities of data. One of the drawbacks of commercial offerings is the large price associated with them, both in terms of purchasing and in terms of running them.

Then, you have your open-source offerings. Currently there are three big names in the open-source reporting realm: JasperReport, Pentaho, and BIRT. Two of these projects, JasperReports and BIRT, are run by commercial companies who make their money by doing professional services for these offerings to small scale and private projects. Again, there are a number of pros and cons associated with open-source solutions. With open-source, you have full access to the source code of the platform you choose. This allows you to add in functionality, embed it with your existing applications, and actively participate in a development community that is oftentimes very large and around the world. There is little initial cost to open-source software in terms of purchasing, as open-source is free. The cons are that there is typically a cost associated with finding individuals who are knowledgeable in open-source.

1.7.3 What is BIRT ?

BIRT (which stands for Business Intelligence and Reporting Tools) is actually a development framework. Adding the word "Tools" to the title acronym is appropriate; BIRT is in fact a collection of development tools and technologies, used

for report development, utilizing the BIRT framework. BIRT isn't necessarily a product, but a series of core technologies that products and solutions are built on top of, similar in fashion to the Eclipse framework.

1.7.4 Why BIRT?

As mentioned before, there are other open-source reporting platforms out there.So what makes BIRT stand out over JasperReports or Pentaho? With JasperReports, in reality, it comes down to tastes. I prefer the "What You See Is What You Get" (WYSIWYG) designer of BIRT over JasperReports. I also like the fact that it does not require compiling of reports to run, and that it is an official Eclipse project. However, that is not to say Jasper is not without strengths of its own. Jasper does do pixel-perfect rendering of reports, which is something that BIRT does not do. Later versions of Jasper also support the Hibernate HQL language.

1.8 MySQL

1.8.1 Introduction

In today's interconnected world, it's almost impossible to find a business that doesn't depend on information in some form or another. Be it marketing data, financial movements, or operational statistics, businesses today live or die by their ability to manage, massage, and filter information flow in order to achieve a competitive advantage.

More often than not, all this data finds a home in a business' relational database management system (RDBMS), a software tool that assists in organizing, retrieving, and cross-referencing information. A large number of such systems are currently available, and you've probably already heard of some of them: Oracle, Sybase,Microsoft Access, and PostgreSQL are well-known names. These database systems are powerful, feature-rich software applications, capable of organizing and searching millions of records at high speeds; as such, they're widely used by businesses and government offices, often for mission-critical purposes.

Recently, though, more and more attention has focused on a relatively new entrant in this field: MySQL. MySQL is a high-performance, multithreaded, multiuser RDBMS built around a client-server architecture. Over the last few years, this fast, robust, and user-friendly database system has become the de facto choice for both business and personal use, notably on account of its advanced suite of data management tools, its friendly licensing policy, and its worldwide support community of users and engineers. This introductory chapter will gently introduce you to the world of MySQL by taking you on a whirlwind tour of MySQL's history, features, and technical architecture.

1.8.2 Top Reasons to Use MySQL

Scalability and Flexibility : The MySQL database server provides the ultimate in scalability, sporting the capacity to handle deeply embedded applications with a footprint of only 1MB to running massive data warehouses holding terabytes of information.

High Performance : A unique storage-engine architecture allows database professionals to configure the MySQL database server specifically for particular applications, with the end result being amazing performance results. With high-speed load utilities, distinctive memory caches, full text indexes,

and other performance-enhancing mechanisms, MySQL offers all the right ammunition for today's critical business systems.

Web and Data Warehouse Strengths : MySQL is the de-facto standard for high-traffic web sites because of its high-performance query engine, tremendously fast data insert capability, and strong support for specialized web functions like fast full text searches. These same strengths also apply to data warehousing environments where MySQL scales up into the terabyte range for either single servers or scale-out architectures.

Comprehensive Application Development : One of the reasons MySQL is the world's most popular open source database is that it provides comprehensive support for every application development need. Within the database, support can be found for stored procedures, triggers, functions, views, cursors, ANSI-standard SQL, and more. MySQL also provides connectors and drivers (ODBC, JDBC, etc.) that allow all forms of applications to make use of MySQL as a preferred data management server. It doesn't matter if it's PHP, Perl, Java, Visual Basic, or .NET, MySQL offers application developers everything they need to be successful in building database-driven information systems.

Management Ease : MySQL offers exceptional quick-start capability with the average time from software download to installation completion being less than fifteen minutes. This rule holds true whether the platform is Microsoft Windows, Linux, Macintosh, or UNIX. Once installed, self-management features like automatic space expansion, auto-restart, and dynamic configuration changes take much of the burden off already overworked database administrators. MySQL also provides a complete suite of graphical management and migration tools that allow a DBA to manage, troubleshoot, and control the operation of many MySQL servers from a single workstation.

Open Source Freedom: Pure awesomeness !

Chapter 2

Architecture

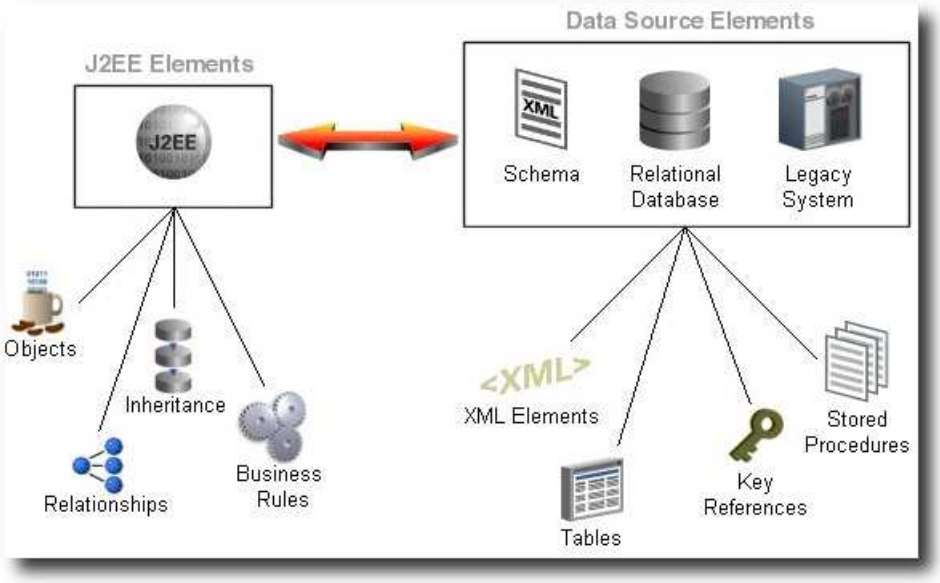

Figure 2.1: Solving Object-Persistence Impedance Mismatch

2.1 Application Server

2.2 JEE

2.2.1 Entities

2.2.2 Persistence

Object-Persistence

Java-to-data source integration is a widely underestimated problem when creating enterprise Java applications. This complex problem involves more than simply reading from and writing to a data source. The data source elements include tables, rows, columns, and primary and foreign keys. The Java and Java EE elements include entity classes, business rules, complex relationships, and inheritance. In a nonrelational data source, you must match your Java entities with EIS records or XML elements and schemas. These differences (as shown in the following figure) are known as the object-persistence impedance mismatch.

What Is Java Persistence API?

The Java Persistence API (JPA) is a lightweight framework for Java persistence based on Plain Old Java Object (POJO). JPA is a part of EJB 3.0 and 3.1 specifications. JPA provides an object relational mapping approach that lets you declaratively define how to map Java objects to relational database tables in a standard, portable way. You can use this API to create, remove and query across lightweight Java objects within both an EJB 3.0/3.1-compliant container and a standard Java SE 5/6 environment.

JPA includes the following concepts:

- Entity–any application-defined object with the following characteristics can be an entity:
  - it can be made persistent
  - it has a persistent identity (a key that uniquely identifies an entity instance and distinguishes it from other instances of the same entity type. An entity has a persistent identity when there is a representation of it in a data store)
  - it is partially transactional in a sense that a persistence view of an entity is transactional (an entity is created, updated and deleted within a transaction, and a transaction is required for the changes to be committed in the database). However, in memory entities can be changed without the changes being persisted.
  - it is not a primitive, a primitive wrapper, or built-in object. An entity is a fine-graned object that has a set of aggregated state that is typically stored in a single place (such as a row in a table), and have relationships to other entities.
- Entity metadata–describes every entity. Metadata could be expressed as annotations (specifically defined types that may be attached to or place in front of Java programming elements) or XML (descriptors).
- Entity manager–enables API calls to perform operations on an entity. Until an entity manager is used to create, read, or write an entity, the entity is just a regular nonpersistent Java object. When an entity manager obtains a reference to an entity, that entity becomes managed by the entity manager. The set of managed entity instances within an entity manager at any given

time is called its persistence context–only one Java instance with the same persistent identity may exist in a persistence context at any time.

You can configure an entity manager to be able to persist or manage certain types of objects, read or write to a particular database, and be implemented by a specific persistence provider. The persistence provider supplies the backing implementation engine for JPA, including the EntityManager interface implementation, the Query implementation, and the SQL generation.Entity managers are provided by an EntityManagerFactory. The configuration for an entity manager is bound to the EntityManagerFactory, but it is defined separately as a persistence unit. You name persistence units to allow differentiation between EntityManagerFactory objects. This way your application obtains control over which configuration to use for operations on a specific entity. The configuration that describes the persistence unit is defined in a persistence.xml file.

The following description expresses relationships between JPA concepts:

- Persistence creates one or more EntityManagerFactory objects
- each EntityManagerFactory is configured by one persistence unit
- EntityManagerFactory creates one or more EntityManager objects
- One or more EntityManager manages one PersistenceContext

EclipseLink

EclipseLink is an advanced, object-persistence and object-transformation framework that provides development tools and run-time capabilities that reduce development and maintenance efforts, and increase enterprise application functionality. Using EclipseLink, you can integrate persistence and object-transformation into your application, while staying focused on your primary domain problem. EclipseLink addresses the disparity between Java objects and data sources. EclipseLink is a persistence framework that manages relational, object-relational data type, EIS, and XML mappings in a seamless manner. This lets you rapidly build applications that combine the best aspects of object technology and the specific data source. EclipseLink lets you do the following:

- Persist Java objects to virtually any relational database supported by a JDBC-compliant driver.

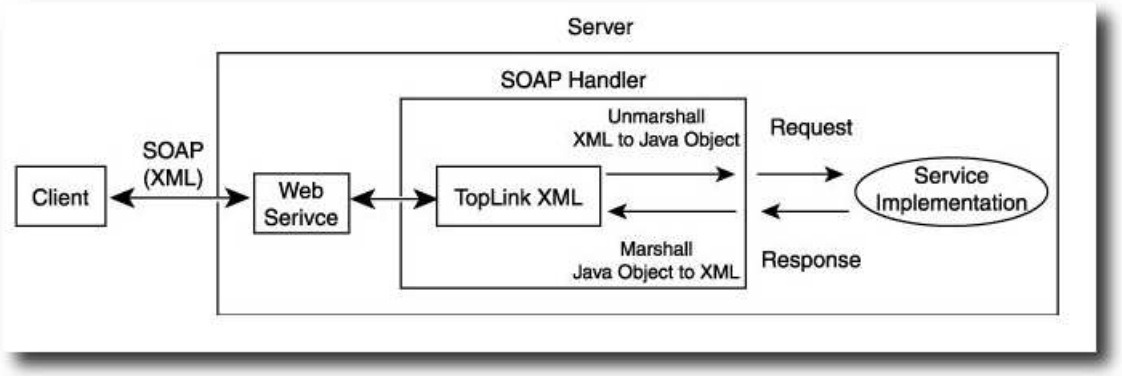

Figure 2.2: Web Services Architecture

- Persist Java objects to virtually any nonrelational data source supported by a Java EE Connector architecture (JCA) adapter using indexed, mapped, or XML enterprise information system (EIS) records.
- Perform in-memory conversions between Java objects and XML Schema (XSD) based XML documents using JAXB.
- Map any object model to any relational or nonrelational schema, using EclipseLink and JPA integration with several tools including Eclipse Dali, Oracle JDeveloper, or NetBeans. The Workbench, a graphical mapping tool, is also provided for backward compatibility for projects using the EclipseLink native API.

2.3 Web services

2.3.1 Considering Web Services Architecture

A Web services architecture is similar to the Three-Tier Architecture or Session Bean Architecture architecture, however, in a Web services architecture, you encapsulate business logic (the service) in a Web service instead of (or in addition to) using session beans. In a Web services architecture, clients communicate with your application using SOAP messages (XML over HTTP).

As in any architecture, you can use EclipseLink to persist objects to relational or EIS data sources. However, in a Web services architecture, you can also use EclipseLink to map your object model to an XML schema for use with the Web service or as the Web service XML serializer.

Chapter 3

Development process

3.1 Java application server setup

3.1.1 Getting Glassfish

GlassFish can be downloaded from GlassFish home. To download it, simply click on the link for your platform, The file should start downloading immediately. After the file finishes downloading, we should have a file called something such as glassfish-v3-unix.sh, glassfish-v3-windows.exe, or glassfish-v3.zip. The exact filename will depend on the exact GlassFish version and platform.

3.1.2 Installing Glassfish server

Installing GlassFish is an easy process. However, GlassFish assumes some dependencies are present on your system.

GlassFish dependencies

In order to install GlassFish 3, a recent version of the Java Development Kit (JDK) must be installed on your workstation (JDK 1.6 or a newer version required), and the Java executable must be in your system path. The latest JDK can be downloaded from `http://java.sun.com/`. Please refer to the JDK installation instructions for your particular platform at:
`http://java.sun.com/javase/6/webnotes/install/index.html`.

Performing the installation

Once the JDK has been installed, installation can begin by simply executing the downloaded file (permissions may have to be modified to make it executable):
./glassfish-v3-unix.sh
The actual filename will depend on the version of GlassFish downloaded. The following steps need to be performed in order to successfully install GlassFish:

1. After running the previous command, the GlassFish installer will start initializing.

2. After clicking Next, we are prompted to accept the license terms.

3. The next screen in the installer prompts us for an installation directory. The installation directory defaults to a directory called glassfishv3 under our home directory. It is a reasonable default, but we are free to change it.

4. The next page in the installer allows us to customize Glassfish's administration and HTTP ports. Additionally, it allows us to provide a username and password for the administrative user. By default, no username and password combination is required to log into the admin console. This default behavior is appropriate for development boxes. We can override this behavior by choosing to provide a username and password in this step in the installation wizard.

5. At this point in the installation, we need to indicate if we would like to install the GlassFish update tool. The update tool allows us to easily install additional GlassFish modules. Therefore, unless disk space is a concern, it is recommended to install it. If we access the internet through a proxy server, we can enter its host name or IP address and port at this point in the installation.

6. Now, we are prompted to either select an automatically detected Java SDK or type in the location of the SDK. By default, the Java SDK matching the value of the JAVA_HOME environment variable is selected.

7. After installation finishes, we are asked to register our copy of GlassFish. At this point, we can link our GlassFish installation to an existing Sun Online Account, create a new Sun Online Account or skip installation.

Verifying the installation

To start GlassFish, change the directory to [glassfish installation directory]/glassfishv3/bin and execute the following command:

```
./asadmin start-domain domain1
```

A few seconds after executing the previous command, we should see a message similar to the following at the bottom of the terminal:

```
Name of the domain started: [domain1] and
its location: [/home/chedi/Application/glassfishv3/glassfish/domains/HyperPath].
Admin port for the domain: [4848].
We can then open a browser window and type the following URL in the browser's
location text field: http://localhost:8080.
```

3.1.3 Installing MySql Database JDBC connector

3.1.4 Adding the JDBC resources

3.1.5 Tuning security

3.2 Database setup

The MySQL database has become the world's most popular open source database because of its high performance, high reliability and ease of use.

3.2.1 Installing the MySql server

3.2.2 Creating the database schema

3.3 Eclipse IDE setup

Eclipse is a popular open source Integrated Development Environment (IDE) that is widely adopted in the developer community.

3.3.1 Installing Eclipse

Because we are both using Fedora Linux operating system, we do not require external resources to install the Eclipse IDE as it came bundled in the install DVD. Nevertheless, you need to select the following package in ordre to get it up and running: [1]

1. eclipse-changelog
2. eclipse-jdt
3. eclipse-rcp
4. eclipse-egit
5. eclipse-pde
6. eclipse-birt
7. eclipse-dtp
8. eclipse-swt

You can install them by using the command:

```
yum install [package name]
```

Or via the graphic interface for adding packages. In the case, you are not using Fedora or any other Linux variant you could always download the Eclipse application from the eclipse.org site. Exactly you need to download the **"Eclipse IDE for Java EE Developers"** available at this url: http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/heliossr2 [2] [3]

[1] Some extra packages will be detected by the install system, but those listed here are the must have to compile and run this project

[2] This URL is relative to the current version of eclipse as, so make sure to go to the correct URL if it get updated

[3] This version include almost all the plugins needed to compile and run the application, we recommend you to add the git plugin to be able to download the code from GitHub

3.3.2 GlassFish plugin

The GlassFish Tools Bundle for Eclipse is a software bundle that includes GlassFish Server v2.1 and v3 Prelude, Eclipse IDE 3.4 and the plug-in that enables them to work together. It provides a ready-to-use environment that helps you get started with developing applications on Eclipse and deploying them to GlassFish Server.

To Install on Linux Platform:

1. Go to ***Help*** → ***Install*** new software
2. Add new repository :
   `http://dlc.sun.com.edgesuite.net/glassfish/eclipse/helios`
   to the Available software sites and name it GlassFish or any convenient name. [4]
3. Select the newly created site from the "Work with" combobox. You should see them the GlassFish bundle.
4. Select the GlassFish plugin and continue to the end of the installation using next to the end.
5. Restart the IDE.

Create a new GlassFish Server

The GlassFish Tools Bundle for Eclipse, by default, installs a new instance of GlassFish v2.1 Server and GlassFish v3 Prelude Server for you. If you want to add a new Server to Eclipse IDE, use the following procedure:

1. Select Servers view.
2. Select New, from Server popup menu. New Server page is displayed
3. Provide GlassFish Server details.
4. Click Next and complete Server configuration. For example, provide values for the following:
   - Domain Directory

[4] don't forget to change the last part of the url to your version of eclipse

- Domain Name
- Administrator ID
- Administrator password

5. Click Next. Add and Remove Projects page is displayed. Select the existing projects, you want to add to the new Server.
6. Click Finish.

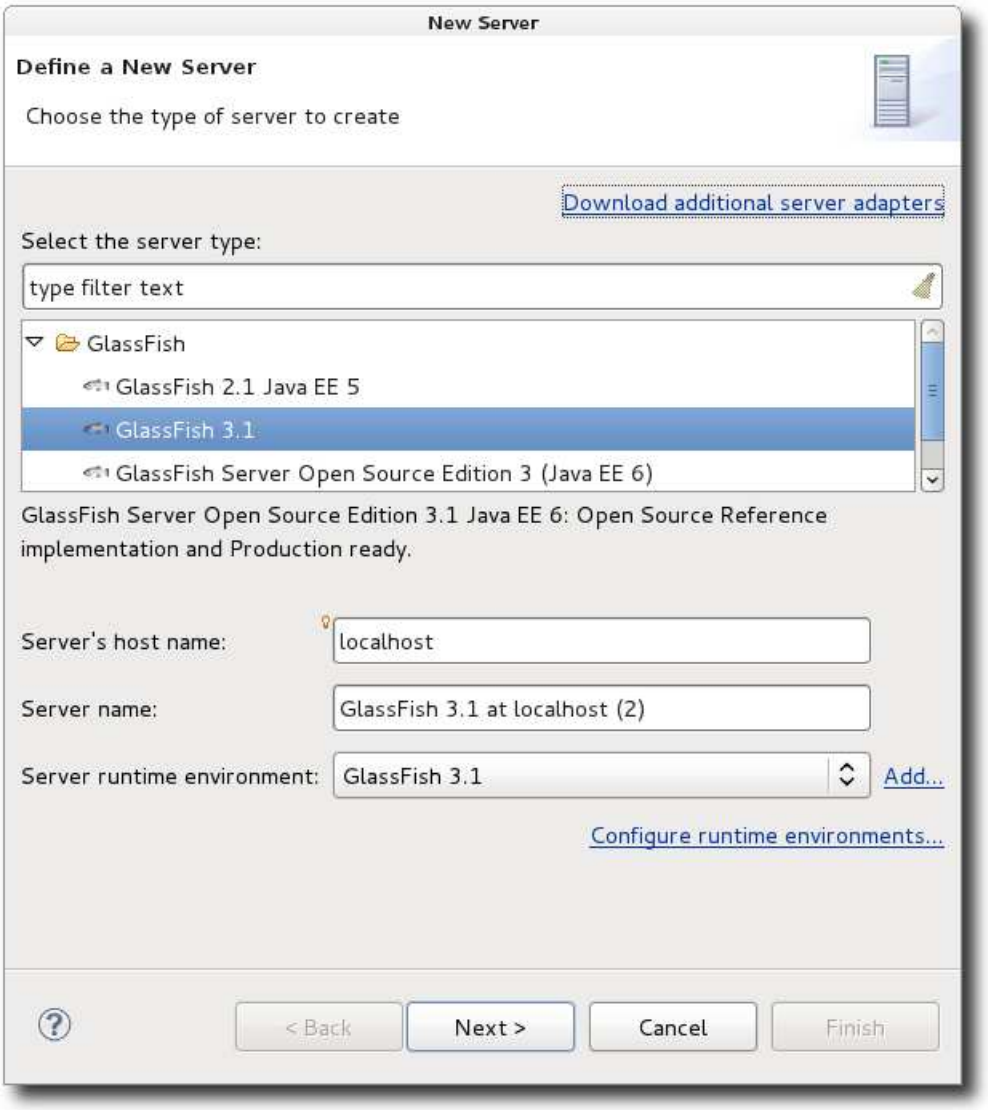

Figure 3.1: Adding new server in Eclipse

To publish and clean Projects

To publish applications to the GlassFish Server, use the following procedure:

1. Select Servers view.
2. Select GlassFish Server.
3. Select Publish from Server popup menu.

To discard all published projects, and republish from scratch, use the following procedure:

1. Select Servers view.
2. Select GlassFish Server.
3. Select Clean... from Server popup menu.
4. Click OK to accept the warning regarding discarding the published state.

To access GlassFish Server Administration Console

GlassFish Sever offers many administrative features that are not accessible from Eclipse IDE. These features are available from GlassFish Server Administration Console which is accessible from Eclipse IDE. The following procedure describes the process of accessing GlassFish Server Administration Console, from Eclipse IDE.
[5]

1. Select Servers view.
2. Select Server popup menu → GlassFish Enterprise Server →GlassFish Administration Console.

[5]The administration console could be accessed via any navigator using the url http://localhost:4848 (assuming that your GlassFish server is installed on the local machine, you could change it to the adequate server name if accessing from an outside location)

3.3.3 Creating the Eclipse Projects

The container project (EAR project)

There are different Eclipse project types that can be used with the JEE 6. In the most flexible setup, you use a so-called Enterprise Application Project, also called an EAR Project, after the artifact it produces, an Enterprise Archive or EAR.

An EAR project is nothing but a container, intended to bundle the artifacts of the contained projects, along with any libraries that they use and that are not already part of the Java Enterprise Edition. We always begin with an EAR project, and then we create those projects, where we do the actual work, adding each of them to the EAR. Later, for deployment, we can export the EAR project as one single EAR file that contains all JARs from the contained projects.

In the JEE perspective you have the Project Explorer on the left side. If you have a fresh workspace, then the Project Explorer will be empty. Either from the File menu or from the Project Explorer context menu choose New / Enterprise Application Project. This opens the New EAR Application Project dialog.

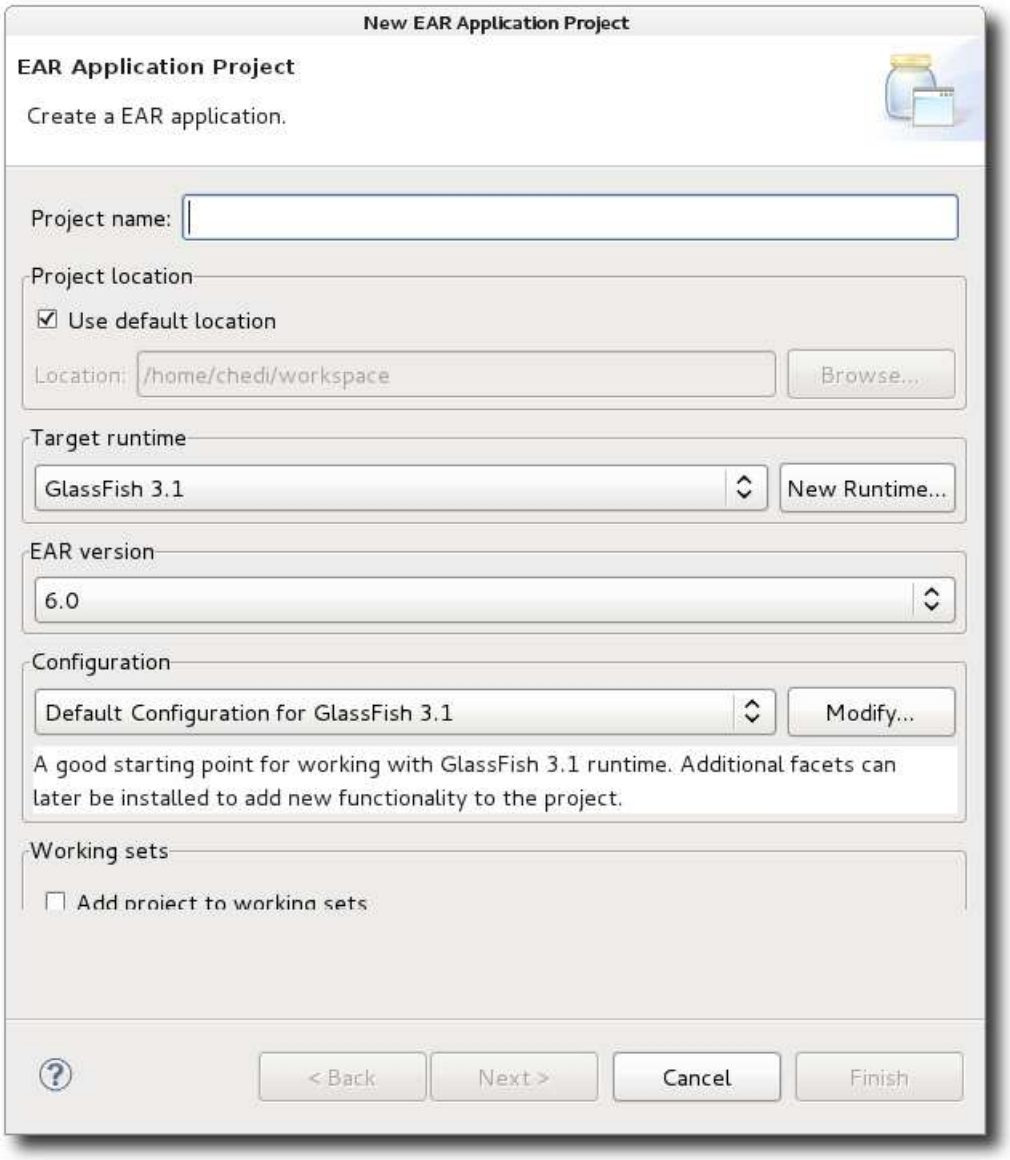

Figure 3.2: Creating Eclipse entreprise project

The EJB project

Enterprise Java Beans (EJB) are really just normal Java classes. It is when EJBs are instantiated under the control of an EJB container, that all sorts of interesting things begin to happen automatically.

For instance you can leave field variables in your beans uninitialized, and as long as they have the right annotations, the container injects proper instances, and for performance reasons it can do that from a pool of pre-allocated, momentarily idle instances.

One of the most important aspects of EJB containers is, that they can inject datasources (for instance JDBC datasources) and other resources, that the program only references by name, and that are really defined in the application server.

When you use JPA, an Entity is again a simple Java class with getters and setters for its fields, a POJO. You make it an entity by doing two things: annotating it with JPA annotations, and letting an EntityManager fetch it from or persist it to the database. You get such an EntityManager again via injection, when you get it, it is already associated with a persistence context that manages transactions and caches entities.

Chapter 4

Implementation

4.1 Data model

4.2 EJB Server Bean

4.2.1 Annotations

Introduction

Annotation is a simple, expressive means of decorating Java source code with metadata that is compiled into the corresponding Java class files for interpretation at run time by a JPA persistence provider to manage persistent behavior.

A metadata annotation represents a Java language feature that lets you attach structured and typed metadata to the source code. Standard JPA annotations are in the javax.persistence package.

Advantages and Disadvantages of Using Annotations

Using annotations provides several advantages:

- They are relatively simple to use and understand.
- They provide in-line metadata within with the code that it describes; you do not need to replicate the source code context of where the metadata applies.

The primary disadvantage of annotations is that the metatdata becomes unnecessarily coupled to the code. changes to metadata require changing the source code.

4.2.2 Abstraction of data access

Mapping relational databases to objects is a well-understood task by now, thus there are plenty of supporting tools, and normally those tools generate code. Eclipse itself comes with a tool to create entities from tables, so do other IDEs and of course the major UML tools can do that as well.

On the other hand, when you access the database via JPA, you either ask the EntityManager for entities that you have an id for, or you create query objects, execute them and get lists of objects as results. If you use queries (which we do), you express them in the Java Persistence Query Language.

4.2.3 Using EclipseLink JPA Extensions for Mapping

The Java Persistence API (JPA), part of the Java Enterprise Edition 5 (Java EE 5) EJB 3.0 specification, greatly simplifies Java persistence and provides an object relational mapping approach that allows you to declaratively define how to map Java objects to relational database tables in a standard, portable way that works both inside a Java EE 5 application server and outside an EJB container in a Java Standard Edition (Java SE) 5 application.

EclipseLink persistence provider applies the first annotation that it finds; it ignores other mapping annotations, if specified. If EclipseLink persistence provider does not find any of the mapping annotations from the preceding list, it applies the defaults defined by the JPA specification: not necessarily the @Basic annotation.

Here we use various annotation to configure the persistent behavior of your entities:

Listing 4.1 : Emails JPA entity

```
@Entity
@XmlRootElement

@Table(
  name    = "emails",
  schema  = "",
  catalog = "hyperPath",

  uniqueConstraints = {
    @UniqueConstraint(columnNames = {"address"})
  })

@NamedQueries({
  @NamedQuery(
    name  = "Emails.findAll",
    query = "SELECT e FROM Emails e"),
  @NamedQuery(
    name  = "Emails.findById",
    query = "SELECT e FROM Emails e WHERE e.id = :id"),
  @NamedQuery(
    name  = "Emails.findByAddress",
    query = "SELECT e FROM Emails e WHERE e.address = :address")
})

public class Emails implements Serializable {
  private static final long serialVersionUID = 1L;
  @Id
  @NotNull
  @Basic(optional = false)
  @Column(name = "id", nullable = false)
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private Integer id;

  @NotNull
```

```
35    @Basic(optional = false)
36    @Size(min = 1, max = 100)
37    @Column(name = "address", nullable = false, length = 100)
38    private String address;
39
40    @ManyToMany(mappedBy = "emailsList")
41    private List<Entities> entitiesList;
42
43     /**
44      * Rest of the Email entity class (POJO) See Emails.java
45      * .......
46      */
47  }
```

4.2.4 JPA Persistence Unit

A persistence unit configures various details that are required when you acquire an entity manager. You specify a persistence unit by name when you acquire an entity manager factory. You configure persistence units in JPA persistence descriptor file *persistence.xml*.

In this file, you can specify the vendor extensions that this reference describes by using a *<properties>* element.

Listing 4.2 : Persistence unit persistence.xml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <persistence version="2.0"
4    xmlns="http://java.sun.com/xml/ns/persistence"
5    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
6    xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
7    http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd">
8
9    <persistence-unit
10     name="HyperPathServerPU"
11     transaction-type="JTA">
12     <provider>
13       org.eclipse.persistence.jpa.PersistenceProvider
14     </provider>
15
16     <jta-data-source>
17       jdbc/HyperPathData
18     </jta-data-source>
19
20     <exclude-unlisted-classes>
21       false
22     </exclude-unlisted-classes>
23
24     <properties>
25       <property
26         name="eclipselink.ddl-generation"
```

```
        value="create-tables"/>
    </properties>

  </persistence-unit>
</persistence>
```

[1]

4.2.5 Calling the bean as a SOAP web service

There is an easy way to access Service Beans: we simply annotate their class as web service. If we don't care about the exact schema of the WDSL that gets generated, it is enough to use one simple annotation like this:

Listing 4.3 : WebService Bean

```
@WebService(serviceName = "EmailsServices")
@Stateless()
public class EmailsServices {
  @Resource
  private UserTransaction utx;

  @PersistenceUnit
  EntityManagerFactory emf;

  /**
   * List all emails
   */
  @WebMethod(operationName = "listAllEmails")
  public List<Emails> listAllEmails() throws
    Exception,
    NonexistentEntityException,
    RollbackFailureException
  {
    emf = Persistence.createEntityManagerFactory("HyperPathServerPU");
    EmailsJpaController controller = new EmailsJpaController(utx, emf);
    return controller.findEmailsEntities();
  }

  /**
   * Other web services, see EmailsServices.java
   * .....
   */
}
```

[1] Avoiding Clear text Passwords: EclipseLink does not support storing encrypted passwords in the persistence.xml file. For a Java EE application, you do not need to specify your password in the persistence.xml file. Instead, you can specify a data-source. This datasource is specified on the application server, and can encrypt the your password with its own mechanism.

Chapter 5

Further work

Chapter 6

Conclusion

Bibliography

[1] David Heffelfinger. *Java EE 6 with GlassFish 3 Application Server*. Packt Publishing, 2010.

[2] Patrick W. Daly Helmut Kopka. *A Guide to LaTeXand Electronic Publishing*. Addison Wesley, Massachusetts, 2004.

[3] Sun Microsystems. *GlassFish Tools Bundle for Eclipse User Guide*, 2009.

[4] Vikram Vaswani. *MySQL Database Usage and Administration*. The McGraw-Hill Companies, 2009.

[5] wikipidia. Geo marketing. http://en.wikipedia.org/wiki/Geo(marketing), 2011.

[6] wikipidia. Location-based service. http://en.wikipedia.org/wiki/Location-based_service, 2011.

Index