Betty project documentation

Version 1.1.0 - February 28th, 2009

License

Audience

The intended audience of this documentation is the person in charge of the server administration.

The administrator is not required any developer's skills: however Windows or Linux administration notions are quite recommended. The usual administrative tasks involve browsing Web pages or using the Adobe Flash client (which is fairly simple), editing configuration files, and invoking scripts (Windows batch files or Linux shell-scripts).

Table of Contents

Installing Betty

Please first ensure that these prerequisites are met before installing Betty: they are really required.

Prerequisites

The following tools are not present in the binary distribution. You have to download and install them all individually, mostly because of their size. All other libraries and tools needed to run Betty are already located in the binary distribution (e.g. the Castor framework, Java Service Wrapper, etc.).

Windows prerequisites

Sun JRE

Download and install the Sun Java Runtime Environment (JRE) 6 from the following Web page: http://java.sun.com/javase/downloads

Linux prerequisites

Sun JRE

Download the Sun Java Runtime Environment (JRE) 6 from the following Web page: http://java.sun.com/javase/downloads

Install the JRE for example in directory "`/usr/java/jre1.6.0_xx`", where "`xx`" must be replaced by the appropriate update number (`12` at the time of writing).

Set the following system environment variables:

```
JAVA_HOME='/usr/java/jre1.6.0_xx'
PATH=$PATH:$JAVA_HOME/bin
```

The installation itself

Please note that Betty uses the JBoss configuration named "**default**", and this configuration cannot be currently changed. The mode "minimal" couldn't be an option, as the project uses EJB2 entities. But the mode "all" would be theoretically possible.

Windows install

Remember that all steps described in "Windows prerequisites" shall be performed first.

The preferred installation mode on Windows platforms is through the installation wizard: "`Betty-1.1.0-x86-32.exe`". However, using the Linux mode described just below is also possible.

You must have administrative rights on your Windows machine in order to install Betty.

The Windows registry is not used by the project, except for some installer keys (when the installation wizard is used), and the Windows service registration.

Linux install

Remember that all steps described in "Linux prerequisites" shall be performed first.

The sole current installation mean on Linux is through a compressed file, "`Betty-1.1.0-x86-32.zip`".

After the installation

Windows post-install

If you installed Betty through the compressed file ("`Betty-1.1.0-x86-32.zip`"), you have to manually register the associated Windows service. Browse to the folder "`bin`", and execute (double-click on) the Windows batch file "`install-service.bat`". You must have administrative rights on your Windows machine in order to execute this script.

Linux post-install

localhost

Ensure that your "`/etc/hosts`" file contains the following lines:

```
127.0.0.1   localhost
xxx.xxx.xxx.xxx   your-server-name
```

Of course replace "`xxx.xxx.xxx.xxx`" by your server's IP address.
You must have administrative rights on your Linux machine in order to edit this file.

Wget

If you plan to use shell-scripts to perform some administrative tasks (like XML configuration files ingest or championship results ingest), the **Wget** tool (www.gnu.org/software/wget) must be properly installed in your Linux distribution. This is highly recommended.

Service

Installing Betty to start when the system boots (and to stop when the system shutdowns) is platform-specific. You will find a good starting point in file "`bin/install-service.sh`". But be aware that you may have to customize it, or even to rewrite it completely. Check also "`bin/uninstall-service.sh`". You must have administrative rights on your Linux machine in order to execute such a script.
In all cases, you have to edit the file "`conf/wrapper.conf`", and replace the two following lines:

- Fix the line "`wrapper.working.dir=.`" at the beginning of the file with the absolute path to your Betty distribution: "`wrapper.working.dir=/usr/java/Betty`" for example.
- Fix the line "`wrapper.java.command=java`" at the beginning of the file with the absolute path to the "`java`" command:
  "`wrapper.java.command=/usr/java/jre1.6.0_12/bin/java`" for example.

Uninstalling Betty

Windows

If you installed Betty from the installation wizard, use the uninstaller created for this purpose:

- either from the Windows “Start / Programs / Betty“ program group (if you renamed the program group at installation time, use the correct name),
- or from the “Add / Remove Programs” of the Windows Control Panel.

The following directory may remain after uninstall: “`server/default/data`”, which you should leave intact if you want to keep the database.

All other files and/or directories that you added after installation will also remain in your Betty distribution.

If you installed Betty from the raw archive (“`Betty-1.1.0-x86-32.zip`”):

- Stop the server if it is running (see “Starting and stopping the server“),
- Remove the Windows service (“`bin\uninstall-service.bat`”),
- Delete your Betty distribution.

Linux

- Stop the server if it is running (see “Starting and stopping the server“),
- Remove the service (platform-specific, see “`bin/uninstall-service.sh`” and see also “Linux post-install“),
- Delete your Betty distribution.

Configuration

Web (HTTP) port

The default HTTP port number of the JBoss Web server is "`8080`". If you want to change this setting, edit the file "`server/default/deploy/jbossweb.sar/server.xml`" with a text editor and replace all occurrences of "`8080`" with your new port number. You must restart the server in order to take this update into account.

Please notice that, if changed, the new HTTP port must also be updated:

- In all **scripts** that you may use (Windows batch files or shell-scripts); they are all located in the directory "`conf`".
- In the Flash configuration file "`server/default/deploy/ROOT.war/crossdomain.xml`".

Please note that the Windows installation wizard can do it all for you automatically.

You must be aware that other Web servers may be running onto your machine, and that **port conflicts** may occur. If you suspect such a behavior, consult the log files of the JBoss server (see the "Troubleshooting" section).

Finally, your machine may be hosting a **firewall** which will block the server Web port. The HTTP port described above is the only port needed by Betty for the client remote access. Thus you must configure your firewall in order to allow this server connection.

Web application context

The Web context of the Betty application is "`/betty/`": this means that your betters will have to use the following URL in order to access the welcome HTML page:

http://theserver:8080/betty/

Where "theserver" and "8080" should of course be replaced by appropriate values.

If you feel it'll annoying for them, you can choose to redirect all traffic from the root context to this "betty" context. For that, simply remove the file "`server/default/deploy/ROOT.war/index.html`" (or rename it to "`index.html.org`" for example). The file "`index.jsp`" that we placed there will thus take the responsibility of the root welcome page, and redirect the HTTP traffic to the "betty" context. The new URL will then be:

http://theserver:8080/

Which is of course more convenient. You don't need to restart the server.

Please note that the Windows installation wizard can do it for you automatically.

Users

There are two types of users: **betters** and **administrators**. Of course, a user may endorse both roles. A better may issue bets on a championship's event, while an administrator can't. An administrator manage other users, adds or updates championships (or parts of a championship), and submit championship's event results. There are two separate Web contexts for these two roles: "`/betty/better/`" and "`/betty/admin/`". The base context, "`/betty/`", is open to everyone.

A user is identified through a “login” identifier, which is associated with a password. A password must never be empty.

No personal information about the user is kept in the database or in configuration files.

In a brand new Betty distribution, only one default user is created:

- An administrator named “`root`”, password “`changeit`” (self explanatory, no?). If you remove all administrators, it will be re-created at the next server start-up. If you choose to change its login or password, update also the administration **scripts** that you may use (Windows batch files or shell-scripts). They are all located in the directory “`conf`”.

Add or update users

These are the current means to add or update a user entry:

- Through an **XML** file and the Web API “`/betty/admin/AddXmlServlet`” (or more simply the `addXml` scripts in directory “`conf`”, see also “Database preparation“). Example of such an XML file (the XML declaration is optional):
```
<users>
     <user id=”demo” password=”demo” isbetter=”true”/>
     <user id=”root” password=”changeit” isadmin=”true”/>
</users>
```
The server must be started, as one of its Web API will be invoked. This method allows you to add or update multiple user entries at one time.
- A dedicated **Web API** is available for the administrator, named “`/betty/admin/AddUserServlet`”. It may be invoked in particular through the raw HTML page “`/betty/admin/raw.html`”, or more conveniently through the Flash client. It may also be invoked from a script file (bat or sh), thanks to the Wget tool or equivalent. This method adds or updates a single user entry at one time.
- A better may **self register** himself or herself to the bet server, through the Web API “`/betty/AddBetterServlet`”. This API may be invoked in particular through the raw HTML page “`/betty/raw.html`”, or more conveniently through the Flash client. Please note that the administrator can disable this function, if all user creations or modifications shall be handled by an administrator for example. If desired, the better self-registration is disabled (or re-enabled) by editing the file “`server/default/deploy/betty/betty.war/WEB-INF/web.xml`”: search for a parameter named “`allowSelfRegistration`”, and set its value to “`false`” (or “`true`”). The server must be restarted for this update to take effect.

Remove users

These are the current means to remove a user entry:

- A dedicated **Web API** is available for the administrator, named “`/betty/admin/RemoveUserServlet`”. It may be invoked in particular through the raw HTML page “`/betty/admin/raw.html`”, or more conveniently through the Flash client. It may also be invoked from a script file (bat or sh), thanks to the Wget tool or equivalent. This method removes a single user entry at one time.
- A better may **self unregister** himself or herself from the bet server, through the Web API “`/betty/better/RemoveBetterServlet`”. This API may be invoked in particular through the raw HTML page “`/betty/better/index.html`”, or more conveniently through the Flash client.

Database preparation

Standard entries

Unless you installed the bet server before, and your database is already fully operational, I highly recommend to begin populating the database with the standard entries provided in the distribution: language resources, test users, locations (countries, cities, etc.) and players (countries, cities); by standard I mean independent from a specific championship. The Windows installation wizard can do it for you automatically. These standard definitions are all present in the directory "`conf`", in an XML form:

- "`resources.xml`": default messages in different languages, to be used by the client GUI.
- "`users.xml`": a test better named "`demo`". Customize this file as needed.
- "`locations.xml`": all countries and many big cities where championship's events may occur. I really spent much time on this file ☺
- "`locationplayers.xml`": all countries and many big cities which can represent players (or teams). This file is automatically created from the previous one through an XSL transformation, "location2player.xsl", located in the same folder.

In no way these pre-configured definitions are exhaustive: you are totally allowed to add custom definitions in your database, or to modify existing ones.

In order to manually add these definitions (or update them), run the following commands from the directory "`conf`" (the invocation order is worthless). The server must be running for these commands to successfully complete.

For Windows:

```
conf> addXml resources.xml
conf> addXml users.xml
conf> addXml locations.xml
conf> addXml locationplayers.xml
```

And for Linux:

```
bob% sh addXml.sh resources.xml
bob% sh addXml.sh users.xml
bob% sh addXml.sh locations.xml
bob% sh addXml.sh locationplayers.xml
```

Running the last two commands may take more time, due to the big amount of new entries to create or update in database. But this is done once for all.

Specific championships

At this point, you still miss at least one championship definition, allowing betters to bet on score results, which is really the goal of this project.

For that, you need to ingest separate XML configuration files, one per championship (example: "`conf/euro-2008.xml`"). You may either use those already present in the project distribution, or download them from the project Web site (they are generally more up-to-date). Unless of course you want to create your own championship, which point is addressed in section "Creating your own championship".

You should also refer to the section "The bet processing model" for the configuration of the **bet model** (namely "how betters are allowed to bet").

Place the championship XML file preferably in the directory "`conf`" of the project distribution (but in fact you can put it wherever you want).

In order to add or update this championship definition, proceed the same way as for the standard definitions described in previous section. The server must be running for this commands to complete successfully.

For Windows:

```
conf> addXml championship.xml
```

And for Linux:

```
bob% sh addXml.sh championship.xml
```

Add a HTTPS Web port

Connecting to the bet server using the HTTPS protocol may sometimes be desirable: the better and administrator restricted areas are protected by a HTTP Basic Authentication method, which means that the password transits almost in clear on the line (Base64 encoding).

The default Web server technology embedded in JBoss is the Apache Tomcat server: it means that configuring a HTTPS connection is in fact adding an HTTPS connector to Tomcat (see in particular http://tomcat.apache.org/tomcat-5.5-doc/ssl-howto.html).

We will describe here the major steps:

1. You need first a **public X.509 certificate** and its associated **private key** for authenticating your server. Concerning the private key, RSA is the preferred algorithm and 1024-bit is the preferred size, according to security experts.
   For that purpose, you may use the tool "`keytool`" present in the JDK (http://java.sun.com/javase/6/docs/technotes/tools/index.html#security), or "OpenSSL" (http://www.openssl.org/). The key point is to use as "Common Name" (CN) the fully qualified host name of your server (in fact, the host name that will be used by your clients to access the Web server). I don't recommend to use an IP address, it may not be understood by your SSL client (I had problems in particular with Java).
2. You should obtain a so-called "key store", with two elements inside it: the public certificate and the private key. This store may be saved in two formats (in order to be parsed by the Java cryptography stack): "JKS" or "PKCS12". JKS has a better support in Java, PKCS12 is more portable to other platforms. If you used keytool, choose JKS. If you used OpenSSL, choose PKCS12.
   And finally at least the private key shall be protected by a password, that we will need later. If you also protect the store with a password, use the same password.
   Place the file in an appropriate folder, like "`conf`".
3. Edit the Tomcat configuration file "`server/default/deploy/jbossweb.sar/server.xml`", and uncomment the SSL connector. Change the HTTPS port number if needed (8443 is the default value, and don't forget to replace all occurrences of 8443 in the file). Add the following attributes into the SSL connector:
   `keystoreFile="${jboss.home.dir}/conf/<store_file_name>"`
   `keystorePass="<the private key password>"`
   `keystoreType="<JKS|PKCS12>"`
4. Edit the Web application descriptor "`server/default/deploy/betty/betty.war/WEB-INF/web.xml`", and change the value of the two "`transport-guarantee`" XML elements from "`NONE`" to "`CONFIDENTIAL`".
5. Restart the server.

Creating your own championship

We will browse now the different notions used in this project.

The **championship** is the root entity; however you may host more than one championship in the server, but they will remain totally isolated. Examples: the FIFA (Football or Soccer) World Cup, the FIA Formula One championship.

A championship defines one or more **phases**: a phase is a group of events, and has a specific meaning in each championship. Examples: in World Cup, a qualification group (A, B, C, D, etc.) is a phase; a quarter-finale is also a phase. Just opposite, the FIA Formula One has only one phase, made of all races. At the end of a phase, we can process final player ranks in this phase: they may be used to define the exact player(s) that will be involved in the phases to come, which were unknown until now. That's essential for the World Cup; but it's (almost ☺) useless for the Formula One championship.

And then a phase defines one or more **events**, the final atomic entity: it occurs in a given location, at a given date and time, and involves two or more **players**. A user **bets** in the scope of an event. In fact, the better issues bets on a player's result in this event. Example: for any confrontation-based event (matches), a better shall usually bet two times, once for each player (even if the graphical interface hides this detail). In the Formula One championship, the server's administrator may have decided that each user shall bet 3 times: for the 3 first race drivers. You understand here the notion of "bet model", which defines how users shall bet, and what scores they will earn for these bets. This point is further described in section "The bet processing model".

The championship structure

A championship is defined in an XML configuration file. You may find in the "`docs`" directory of the binary distribution, the associated DTD (and HTML documentation, "`docs/dtd`") and the associated XML Schema ("`docs/schema`").

Below is a commented version of a (simplified) championship definition:

```
<!-- The championship must be uniquely identified through its "id" attribute -->
<championship id="formula1-2008">

  <!-- Zero, one or more championship's descriptions, for different locales;
       Default locale is "en", or "en_US". -->
  <description>FIA Formula One 2008</description>
  <description locale="fr">Formule 1 2008</description>

  <!-- An optional link to some "official" Web page; purely cosmetic -->
  <link>http://www.formula1.com/</link>

  <!-- An optional list of championship-specific resources.
       They override those defined in conf/resources.xml -->
  <resources>
    <resource id="player">
      <description>Driver</description>
      <description locale="fr">Pilote</description>
    </resource>
  </resources>

  <!-- A championship may define a global score processing model for all its
       events.
       Refer to the appropriate section for more information on this point,
       and see below for an example of a score model configuration. -->

  <!-- A championship may define a global rank processing model for all its
       phases.
```

```
      Refer to the appropriate section for more information on this point,
      and see below for an example of a rank model configuration. -->

  <!-- The definition of the betting model in this championship.
      Refer to the appropriate section for more information on this point -->
  <betmodel id="christophedelory.betty.model.bet.RankBetModel">

    <!-- The description element is the same as the championship's one -->
    <description>Description of the bet model</description>

    <param id="nb-bets">3</param>
  </betmodel>

  <!-- A championship may optionally define here specific locations, in addition
      of those already described in "conf/locations.xml".
      Example:
  <locations>
    <location id="Melbourne" parentid="AU" timezone="Australia/Melbourne"/>
  </locations>
  -->

  <!-- An optional list of players, in addition of those already described in
      "conf/locationplayers.xml". -->
  <players>
    <player id="Hamilton" country="GB">
      <!-- The description element is the same as the championship's one -->
      <description>Lewis Hamilton</description>
    </player>
    <player id="Heidfeld" country="DE">
      <description>Nick Heidfeld</description>
    </player>
  </players>

  <!-- One or more phases in this championship, each of them uniquely identified
      by the "id" attribute.
      This identifier must be unique ONLY in the scope of the enclosing
      championship. -->
  <phase id="2008">

    <!-- A phase may be described by one or more description elements. -->

    <!-- A phase may define a link to a Web page, as in the championship -->

    <!-- A phase may define a global score processing model for all
         its child events.
         Refer to the appropriate section for more information on this point.
    -->
    <scoremodel id="christophedelory.betty.model.score.RankScoreModel">
    </scoremodel>

    <!-- The definition of the rank processing model in this phase.
         Refer to the appropriate section for more information on this point -->
    <rankmodel>
      <param id="comparators">
christophedelory.betty.model.rank.GeneralHighestScoreRankProcessor
christophedelory.betty.model.rank.GeneralLowestResultRankProcessor</param>
    </rankmodel>

    <!-- One or more events in this phase, each of them uniquely identified by
         the "id" attribute.
         This identifier must be unique ONLY in the scope of the enclosing
         phase.
         The date format is the complete date plus hours and minutes format of
         W3C Date-Time (derived from ISO8601). There are two cases:
```

```
          - Without time zone ("YYYY-MM-DD'T'hh:mm"), the time zone is derived
            from the event's location;
          - In the Zulu time zone ("YYYY-MM-DD'T'hh:mm'Z'"). -->
      <event id="AU" date="2008-03-16T15:30">

        <!-- An event may be described by one or more description elements. -->

        <!-- An event shall define a score processing model, unless already done
             in its enclosing phase or championship.
             Refer to the appropriate section for more information on this point,
             and see above for an example of a score model configuration. -->

        <!-- A reference of the event's location. -->
        <location id="Melbourne"/>

        <!-- Two or more players in this event, by reference. -->
        <playerid id="Hamilton"/>
        <playerid id="Heidfeld"/>
      </event>
    </phase>
</championship>
```

The bet processing model

You generally have to define how your betters must bet, and the scores (or points) they will earn for their bets: this is the championship's bet model. The project currently supports 2 bet models, described below.

In a championship XML configuration file, the bet model is defined through the XML element "`championship/betmodel`".

They all share a common configuration parameter, "**`nb-bets`**": the number of bets expected per event (remember that there is one bet per event's player: to be more precise, one bet on each player's result). This number cannot exceed the number of players involved in the event, and must be greater or equal to 1. A better must bet "n" times ("n" being this number), and is allowed not to bet at all (0 times). This number defaults to 1.

Name (identifier)	Description

`christophedelory.betty.model.bet.VersusBetModel`	Based on the result of the confrontation between two players only; points can be earned in two cases:
	• The better guessed the two player's results.
	• The better guessed the event's result, but not the exact player's results.
	The number of points earned depends on 2 configuration parameters:
	`<param id="exact-player-results">`*`points`*`</param>`
	`<param id="exact-event-result">`*`points`*`</param>`
	where *`points`* is of course the number of points earned in the two cases.

`christophedelory.betty.model.bet.RankBetModel`	Based on the difference between the player result (its rank) and the user bet: if the bet is successful (the bet is the player rank), points are earned. The exact number of points depends on the rank itself, according to configuration parameters: `<param id="`*`rank`*`">`*`points`*`</param>` where *`rank`* is a player rank ranging from 1 to n, and *`points`* is the number of points earned for a good bet for this rank.

The default bet processing model is the "`RankBetModel`", with one required bet and no points earned for any rank (which is not very useful ☺).

Other bet models may be added in the future, as the model mechanism is extensible (Java "plug-ins").

The score processing model

When a championship's event has occurred, the administrator must ingest the player's results. From now on, the server needs to know how to rank the players between them in the scope of this event, according to their results. This is necessary first because the betters may be assigned points for these ranks, and second because other (future) events may depend on player ranks in previous events.

We perform the players ranking in the event through player **scores**, assigned thanks to a **score processing model**. The project currently supports the score models described in the table below.

In a championship XML configuration file, the score model is defined through the XML element "`scoremodel`", that may be expressed in 3 different places, even at the same time:

- At the **championship** level: this score model would thus be a default and global score model for all events defined in this championship.
- At the **phase** level: this score model would override the default championship's score model (if any), and act as the new default model for all child events of this phase.
- And finally of course at the **event** level itself.

Name (identifier)	Description

`christophedelory.betty.model.score.VersusScoreModel`	Based on the confrontation of two players only: either one wins and the other looses, or they achieve the same result.
	The score depends on 3 configuration parameters:
	`<param id="win">`*`score`*`</param>`
	`<param id="loose">`*`score`*`</param>`
	`<param id="equal">`*`score`*`</param>`
	where *`score`* is of course the associated player score.

`christophedelory.betty.model.score.RankScoreModel`	Based on the player rank in an event: the player result is a rank, and this player is assigned points (the score) according to this rank. The score depends on the rank according to configuration parameters: `<param id="`*`rank`*`">`*`score`*`</param>` where *`rank`* is a player rank ranging from 1 to n, and *`score`* is the associated player score.
`christophedelory.betty.model.score.ResultScoreModel`	Based on the player result in an event: the score is the result.

The default score processing model is the "`ResultScoreModel`".

Other score models may be added in the future, as the model mechanism is extensible (Java "plug-ins").

The rank processing model

We have seen in previous chapter a mean to rank players in the scope of an event (the score processing model); however we need the same function in the scope of a **phase**: the only reason is that other (future) events in successive phases may depend on player ranks in previous phases. The best example is qualification groups in the World Cup, which will decide the players for the quarter finals; in this case, the qualification group is a phase, and a quarter final is also a phase.

We perform the players ranking in the phase through player **ranks** assigned thanks to a **rank processing model**. The project currently supports only one rank model, described below, which is also of course the default one. There is only one rank model because it's really general-purpose, and can be used for almost all types of phases.

In a championship XML configuration file, the rank model is defined through the XML element "`rankmodel`", that may be expressed in 2 different places, even at the same time:

- At the **championship** level: this phase model would thus be a default and global phase model for all child phases of the championship.
- And of course at the **phase** level itself.

The default and sole rank processing model at this time is named "`christophedelory.betty.model.rank.DefaultRankModel`". It is based on one or more configurable rank processors. These rank processors assign ranks to a given group of players involved in the phase. They are **criteria** which are used to determine the rankings. We will study them below with two examples.

So the unique configuration parameter of the default rank processing model is the list of its rank processors. Example:

```
<rankmodel>
  <param id="comparators">
    christophedelory.betty.model.rank.GeneralHighestScoreRankProcessor
  </param>
</rankmodel>
```

Notes:

1. We do not need to specify the rank model identifier ("`id`" attribute), because there is only one rank model at this time.
2. The **default** list of rank processors for a "`DefaultRankModel`" is composed of only one rank processor:
   "`christophedelory.betty.model.rank.GeneralHighestScoreRankProcessor`"
   (see the examples below for explanations).

Other rank models may be added in the future, as the model mechanism is extensible (Java "plug-ins").

First (simple) example: the Formula One championship

Its definition is the following:

```
<rankmodel>
  <param id="comparators">
    christophedelory.betty.model.rank.GeneralHighestScoreRankProcessor
    christophedelory.betty.model.rank.GeneralLowestResultRankProcessor
  </param>
</rankmodel>
```

The definition order of the rank processors is very important: they are applied in sequence, from the first to the last, until all player ranks have been resolved: as long as two or more players share the same rank. If the first one do the job, perfect! That's all done, and none of the other rank processors will be used. If, after having used the last rank processor, some players still have the same rank, the server cannot perform any further processing, and there are two cases: either this isn't so important for the next phases, and forget it; or the administrator has to perform a manual action: modify the event scores to resolve the issue, or update manually the players for the next phases, or add another criterion at the end of the list.

In this example, both of the rank processors work on the whole list of events in the corresponding phase (prefix "`General`"). We will see in the next example another prefix, "`Local`", which means that only the events involving a given list of players shall be considered for the rank processing (and thus not all phase events, as in the "general" case).

The first rank processor described in the example is based on the sum of the player scores in the phase, the highest being the best. And the second rank processor ranks the players according to the sum of their ranks, the lowest (rank 1) being the best.

So, why all that stuff? Because in a Formula One championship, we rank the drivers first according to the points (i.e. score) they have earned in the season. And if 2 or more drivers reach the same number of points, they are ranked according to their previous race positions (event ranks).

Second (more complicated) example: the UEFA Euro

Its definition is the following:

```
<rankmodel>
 <param id="comparators">
  christophedelory.betty.model.rank.GeneralHighestScoreRankProcessor
```

```
  christophedelory.betty.model.rank.LocalHighestScoreRankProcessor
  christophedelory.betty.model.rank.LocalHighestResultDifferenceRankProcessor
  christophedelory.betty.model.rank.LocalHighestResultRankProcessor
  christophedelory.betty.model.rank.GeneralHighestResultDifferenceRankProcessor
  christophedelory.betty.model.rank.GeneralHighestResultRankProcessor
 </param>
</rankmodel>
```

This is the explanation from the UEFA itself:

“If two or more teams are equal on points on completion of all the matches in their group, the following criteria will be used to determine the rankings in the given order:

a) number of points obtained in the matches among the teams in question;

b) goal difference in the matches among the teams in question:

c) number of goals scored in the matches among the teams in question (if more than two teams finish equal on points);

d) goal difference in all the group matches;

e) number of goals scored in all the group matches;

f) coefficient from the qualifying competitions for the 2006 FIFA World Cup and the 2006/08 UEFA European Football Championship (points obtained divided by the number of matches played);

g) fair play conduct of the teams (final tournament);

h) drawing of lots.”

In our rank processors configuration, the first one corresponds to the “points on completion of all the matches in their group”: the teams are first ranked according to their score in the matches. And then each of the following 5 rank processors correspond to the a, b, c, d and e items from the UEFA. What about f, g and h: if you know how to model it easily in the server, I would be interested; especially for the last 2 ones...

Available rank processors

Here follows a list of all currently available rank processors:

Name (identifier)	Description
`christophedelory.betty.model.rank.GeneralHighestScoreRankProcessor`	Apply to all phase's events. Ranking based on the sum of the player scores in those events, the highest being the best.
`christophedelory.betty.model.rank.GeneralLowestResultRankProcessor`	Apply to all phase's events. Ranking based on the sum of the player results in those events, the lowest being the best.
`christophedelory.betty.model.rank.GeneralHighestResultRankProcessor`	Apply to all phase's events. Ranking based on the sum of the player results in those events, the highest being the best.
`christophedelory.betty.model.rank.GeneralHighestResultDifferenceRankProcessor`	Apply to all phase's events. Ranking based on the sum of the differences between a player's result and all its opponent's results in those events, the highest being the best.
`christophedelory.betty.model.rank.LocalHighestScoreRankProcessor`	Apply to the phase's events involving only a given list of players. Ranking based on the sum of the player scores in those events, the highest being the best.
`christophedelory.betty.model.rank.LocalLowestResultRankProcessor`	Apply to the phase's events involving only a given list of players. Ranking based on the sum of the player results in those events, the lowest being the best.
`christophedelory.betty.model.rank.LocalHighestResultRankProcessor`	Apply to the phase's events involving only a given list of players. Ranking based on the sum of the player results in those events, the highest being the best.
`christophedelory.betty.model.rank.LocalHighestResultDifferenceRankProcessor`	Apply to the phase's events involving only a given list of players. Ranking based on the sum of the differences between a player's result and all its opponent's results in those events, the highest being the best.

Operating the server

Starting and stopping the server

Whatever the host platform, I observed that the server boot time can range from 1 to 2 minutes (from the start command to the complete setup of the server). So be patient.

Windows

If you installed Betty from the installation wizard, you have access to the Windows "Start / Programs / Betty" program group, and can start or stop it from there. If you renamed the program group at installation time, use the correct name.

If you installed Betty from the raw archive ("`Betty-1.1.0-x86-32.zip`"), you can use the Windows batch files located directly in the root installation directory, "`launch.bat`" and "`stop.bat`" (just double-click on the file).

Otherwise, in all cases, you can start or stop the server through the "Services" management console. The service is named "Betty" (internal name "`Betty`").

Remember that the service is **automatically** started at Windows start up. If you prefer to start it manually on demand, open the Windows "Services" management console, and edit the "Betty" service properties (start type on demand).

Linux

In order to start (and stop) the server, you can:

- Use the shell-scripts "`launch.sh`" and "`stop.sh`" present in the root distribution directory,
- Type the following command directly in the root distribution directory:
  `bob% **sh main.sh console**`
  Type <Control-C> to stop the server.
- Install the server as a Linux service, as explained in section "Linux post-install".

If you installed the Betty distribution as root, run the server as root (because of the access permissions).

Submitting event's results

There are currently 2 means for a championship's administrator to submit an event's result:

- A dedicated **Web API** is available for the administrator, named "`/betty/admin/AddPlayerScoreHttpServlet`". It may be invoked in particular through the raw HTML page "`/betty/admin/raw.html`", or more conveniently through the Flash client. It may also be invoked from a script file (bat or sh), thanks to the Wget tool or equivalent. This method adds or updates a single player result at a time. For the whole event, this API must be invoked for each event's player.

- Through the championship **XML** configuration file and the Web API "`/betty/admin/AddXmlServlet`" (or more simply the `addXml` scripts in directory "`conf`", see also "Specific championships").
  In each "`playerid`" element of the target event, add a "`result`" attribute which value is the player's result in this event.
  The server must be started, as one of its Web API will be invoked. This method allows you to add or update multiple player results at one time.

Sometimes you may have to change the processing models in a championship (the bet model of a championship, the rank model of a phase or the score model of an event; see for example "The bet processing model"). In those cases, you have to **reprocess all** player scores and user bets in your championship. A dedicated Web API exists for that purpose: "`/betty/admin/ProcessChampionshipHttpServlet`". It may be invoked through the raw HTML page "`/betty/admin/raw.html`", or more conveniently through the Flash client.

Troubleshooting

First consult the server log files. They are all located in the directory "`server/default/deploy/log`". We present below the different types of log files you may encounter, in the order you should analyze them:

1. "`wrapper.log`": the bootstrap logs. If any error is related to the Java Virtual Machine or to the JBoss base environment, you should find it there. Once the server starts properly, you shouldn't need it anymore.
2. "`server.log`": the most important. An error encountered during the JBoss server lifetime will be logged here. All application errors (including our bet server) can be found in this file.
3. "`localhost_access_log.YYYY-MM-DD.log`" (**not enabled by default**, uncomment the "`AccessLogValve`" in "`server/default/deploy/jbossweb.sar/server.xml`"): all accesses to the Web server are logged here, successful or not. May be useful to debug authentication issues.
4. "boot.log": I rarely use this file. It contains debug information related to the JBoss server startup.
5. In addition, a log file dedicated to the HSQL database is located in directory "`server/default/data/hypersonic`", and is named "`localDB.log`". I do not use it very often, but who knows?

If the log files don't give any clue for the failure cause, then the hard stuff begins. Here follows some hints that may be useful:

- If the server is properly started, a file named "`Betty.pid`" must be present in the root directory of the distribution.
- Use the "raw" HTML version of the Web page to retrieve various information about the bet server ("`/betty/raw.html`").
- The Hypersonic database may be browsed thanks to the following command from the root distribution directory: "`java -cp server/default/lib/hsqldb.jar org.hsqldb.util.DatabaseManager -url jdbc:hsqldb:server/default/data/hypersonic/localDB -user sa`". The server must be stopped for this command to work.
  At run-time, you may launch the same application through the JMX Management Console of JBoss: choose MBean "`database=localDB,service=Hypersonic`" ("`jboss`" section), then invoke operation "`startDatabaseManager()`" on it. You must perform this action on the server itself, as it will launch a Java GUI.

Web API

Here follows the full list of HTTP(S) API exposed by the Betty Web application.

For more information on the HTTP+XML API, the complete DTD may be found in folder "`docs/dtd`" (for a HTML documentation, open "`index.html`"), and the XML Schema in folder "`docs/schema`".

Information API (controls)

Resources list

Description	Returns an XML stream including the list of global language resources (thus excluding the championship-specific resources), currently defined in the server database
Since	0.90
HTTP request path	`/betty/ListResourcesServlet`
Required security role	-
HTTP request body	-
HTTP response	An XML element named "`resources`" and its associated child elements (see the example below)

HTTP request parameters:

Name	Description	Use	Default value	Comment
`locale`[1]	One or more locale identifiers	Optional	HTTP header "`Accept-Language`", otherwise server's default locale	The format is the following: "`language_country_variant`", where `language` is a two-letter ISO-639 code, `country` is a two-letter ISO-3166 code, and `variant` is a vendor or browser-specific code. If the language is missing, the string will begin with an underbar.

Request example:

http://localhost:8080/betty/ListResourcesServlet

Response example (excerpt):

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<resources version="1.1.0">
  <resource id="bet">
    <description>Bet</description>
    <description locale="fr">Pari</description>
  </resource>
</resources>
```

1 This parameter currently has no effect on the returned XML stream

Championships list

Description	Returns an XML stream including the full list of championships, along with the full list of language resources, currently defined in the server database
Since	0.90
HTTP request path	`/betty/ListChampionshipsServlet`
Required security role	-
HTTP request body	-
HTTP response	An XML element named "`championships`" and its associated child elements (see the example below)

HTTP request parameters:

Name	Description	Use	Default value	Comment
`locale`[2]	One or more locale identifiers	Optional	HTTP header "`Accept-Language`", otherwise server's default locale	The format is the following: "`language_country_variant`", where `language` is a two-letter ISO-639 code, `country` is a two-letter ISO-3166 code, and `variant` is a vendor or browser-specific code. If the language is missing, the string will begin with an underbar.

Request example:

http://localhost:8080/betty/ListChampionshipsServlet

Response example (excerpt):

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<championships version="1.1.0">
  <championship id="formula1-2008">
    <link>http://www.formula1.com/</link>
    <description>FIA Formula One 2008</description>
    <description locale="fr">Formule 1 2008</description>
    <resources>
      <resource id="phase">
        <description>Season</description>
        <description locale="fr">Saison</description>
      </resource>
    <resources>
    <betmodel id="christophedelory.betty.model.bet.RankBetModel">
      <param id="nb-bets">3</param>
      <param id="exact-player-results">5</param>
      <param id="exact-event-result">2</param>
    </betmodel>
  </championship>
</championships>
```

2 This parameter currently has no effect on the returned XML stream

Locations list

Description	Returns an XML stream including the full list of locations currently defined in the server database
Since	0.90
HTTP request path	`/betty/ListLocationsServlet`
Required security role	-
HTTP request body	-
HTTP response	An XML element named "`locations`" and its associated child elements (see the example below)

HTTP request parameters:

Name	Description	Use	Default value	Comment
`locale`	One or more locale identifiers	Optional	HTTP header "`Accept-Language`", otherwise server's default locale	The format is the following: "`language_country_variant`", where `language` is a two-letter ISO-639 code, `country` is a two-letter ISO-3166 code, and `variant` is a vendor or browser-specific code. If the language is missing, the string will begin with an underbar. This parameter influences only the "`description`" elements attached to a country (if the location's "`iscountry`" attribute is set to "`true`"), as shown in the example below.

Request example:

http://localhost:8080/betty/ListLocationsServlet?locale=en&locale=fr

Response example (excerpt):

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<locations version="1.1.0">
  <location id="Africa">
    <location country="DZ" iscountry="true" id="DZ">
      <location timezone="Africa/Algiers" country="DZ" id="Algiers" />
      <description locale="en">Algeria</description>
      <description locale="fr">Algérie</description>
    </location>
  </location>
</locations>
```

Players list

Description	Returns an XML stream including the full list of players currently defined in the server database
Since	0.90
HTTP request path	`/betty/ListPlayersServlet`
Required security role	-
HTTP request body	-
HTTP response	An XML element named "`players`" and its associated child elements (see the example below)

HTTP request parameters:

Name	Description	Use	Default value	Comment
`locale`[3]	One or more locale identifiers	Optional	HTTP header "`Accept-Language`", otherwise server's default locale	The format is the following: "`language_country_variant`", where `language` is a two-letter ISO-639 code, `country` is a two-letter ISO-3166 code, and `variant` is a vendor or browser-specific code. If the language is missing, the string will begin with an underbar.

Request example:

http://localhost:8080/betty/ListPlayersServlet

Response example (excerpt):

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<players version="1.1.0">
  <player country="DZ" iscountry="true" id="DZ" />
  <player country="DZ" id="Algiers" />
  <player country="IT" id="Fisichella">
    <description>Giancarlo Fisichella</description>
  </player>
</players>
```

Users list

Description	Returns an XML stream including the full list of users (betters and administrators) currently defined in the server database
Since	0.90
HTTP request path	`/betty/ListUsersServlet`
Required security role	-
HTTP request body	-
HTTP response	An XML element named "`users`" and its associated child elements (see the example below)

HTTP request parameters:

3 This parameter currently has no effect on the returned XML stream

Name	Description	Use	Default value	Comment
`locale`[4]	One or more locale identifiers	Optional	HTTP header "`Accept-Language`", otherwise server's default locale	The format is the following: "`language_country_variant`", where `language` is a two-letter ISO-639 code, `country` is a two-letter ISO-3166 code, and `variant` is a vendor or browser-specific code. If the language is missing, the string will begin with an underbar.

Request example:

http://localhost:8080/betty/ListUsersServlet

Response example:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<users version="1.1.0">
  <user isadmin="true" id="root" />
  <user isbetter="true" id="demo" />
</users>
```

Championship information

Description	Returns an XML stream fully describing a given championship
Since	0.90
HTTP request path	`/betty/GetChampionshipInfoServlet`
Required security role	-
HTTP request body	-
HTTP response	An XML element named "`championship`" and its associated child elements (see the example below)

HTTP request parameters:

4 This parameter currently has no effect on the returned XML stream

Name	Description	Use	Default value	Comment
`championship-id`	The championship identifier	Required	None	-
`locale`	One or more locale identifiers	Optional	HTTP header "`Accept-Language`", otherwise server's default locale	The format is the following: "`language_country_variant`", where `language` is a two-letter ISO-639 code, `country` is a two-letter ISO-3166 code, and `variant` is a vendor or browser-specific code. If the language is missing, the string will begin with an underbar. This parameter influences only the "`description`" elements attached to a country (if the event location's "`iscountry`" attribute is set to "`true`"); see "Locations list".

Request example:

http://localhost:8080/betty/GetChampionshipInfoServlet?championship-id=formula1-2008

Response example (excerpt, with no user bets):

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<championship id="formula1-2008" version="1.1.0">
  <description>FIA Formula One 2008</description>
  <description locale="fr">Formule 1 2008</description>
  <link>http://www.formula1.com/</link>
  <locations>
    <location parentid="Europe" country="FI" iscountry="true" id="FI">
      <description locale="en_US">Finland</description>
    </location>
    <location id="Europe" />
    <location parentid="Europe" country="DE" iscountry="true" id="DE">
      <description locale="en_US">Germany</description>
    </location>
  </locations>
  <players>
    <player country="FI" id="Raikkonen">
      <description>Kimi Räikkönen</description>
    </player>
    <player country="DE" id="Heidfeld">
      <description>Nick Heidfeld</description>
    </player>
  </players>
  <betmodel id="christophedelory.betty.model.bet.RankBetModel">
    <param id="nb-bets">3</param>
    <param id="exact-player-results">5</param>
    <param id="exact-event-result">2</param>
  </betmodel>
  <phase id="2008">
    <rankmodel id="christophedelory.betty.model.rank.DefaultRankModel">
      <param id="comparators">
christophedelory.betty.model.rank.GeneralHighestScoreRankProcessor
christophedelory.betty.model.rank.GeneralLowestResultRankProcessor</param>
    </rankmodel>
```

```
    <playerid score="19" rank="1" result="11" id="Raikkonen" />
    <playerid score="16" rank="2" result="12" id="Heidfeld" />
    <event date="2008-03-16T04:30Z" id="AU">
      <location timezone="Australia/Melbourne" parentid="AU" country="AU"
id="Melbourne" />
      <playerid score="8" rank="2" result="2" id="Heidfeld" />
      <playerid score="1" rank="8" result="8" id="Raikkonen" />
      <scoremodel id="christophedelory.betty.model.score.RankScoreModel">
        <param id="1">10</param>
        <param id="2">8</param>
        <param id="3">6</param>
        <param id="4">5</param>
        <param id="5">4</param>
        <param id="6">3</param>
        <param id="7">2</param>
        <param id="8">1</param>
      </scoremodel>
    </event>
  </phase>
</championship>
```

RSS feed

Description	Returns a RSS stream (v2.0) describing the latest updates or additions in the server database. Deletions are not tracked.
Since	0.92
HTTP request path	`/betty/RSSServlet`
Required security role	-
HTTP request body	-
HTTP response	A RSS 2.0 stream (see the example below). The RSS specification can be found here: http://blogs.law.harvard.edu/tech/rss

HTTP request parameters:

Name	Description	Use	Default value	Comment
`for-admin`	Specifies if the RSS feed is targeted to an administrator or not	Optional	Not set (false)	If set, the resulting RSS feed will include detailed items: all unitary updates or additions to the server database. This may be used for audit. Otherwise, the resulting feed will summarize updates and additions, and is targeted to a simple better.
`max-items`	Number of items to include in the RSS channel	Optional	20	If negative or null, no limit is enforced.

Request example:

http://localhost:8080/betty/RSSServlet?max-items=1

Response example:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<rss version="2.0">
 <channel>
    <title>Betty v1.1.0 better RSS feed</title>
    <link>http://localhost:8080/betty/</link>
    <description>Updated and new items on your Betty server</description>
```

```
    <language>en</language>
    <copyright>Copyright (c) 2008, Christophe Delory</copyright>
    <pubDate>Thu, 12 Jun 2008 15:44:07 +0200</pubDate>
    <lastBuildDate>Thu, 12 Jun 2008 15:31:42 +0200</lastBuildDate>
    <generator>Betty server v1.1.0</generator>
    <docs>http://blogs.law.harvard.edu/tech/rss</docs>
    <image>
      <url>http://localhost:8080/betty/betty.gif</url>
      <title>Betty v1.1.0</title>
      <link>http://localhost:8080/betty/</link>
      <width>64</width>
      <height>64</height>
      <description>Betty v1.1.0</description>
    </image>
    <item>
      <title>New: Bob has bet on event 5, phase A, championship
euro-2008</title>
      <description>Bets: 0 for player CH, 0 for player PT</description>
      <author>Bob</author>
      <guid isPermaLink="false">ejb/betty/LocalBet:1</guid>
      <pubDate>Thu, 12 Jun 2008 15:31:42 +0200</pubDate>
      <source url="http://localhost:8080/betty/RSSServlet">Betty v1.1.0 better
RSS feed</source>
    </item>
  </channel>
</rss>
```

Action API (commands)

Better self-registration

Description	Adds a new better in the server database. This API is enabled only if the configuration parameter "`allowSelfRegistration`" has been set to "`true`"; see "Add or update users".
Since	0.90
HTTP request path	`/betty/AddBetterServlet`
Required security role	-
HTTP request body	-
HTTP response	A HTTP response code only

HTTP request parameters:

Name	Description	Use	Default value	Comment
`id`	The better identifier ("login")	Required	None	
`password`	The better password	Required	None	

Request example:

http://localhost:8080/betty/AddBetterServlet?id=test&password=test

Better password update

Description	Updates the password of the currently logged better.
Since	0.90
HTTP request path	`/betty/better/UpdateBetterServlet`
Required security role	Better
HTTP request body	-
HTTP response	A HTTP response code only

HTTP request parameters:

Name	Description	Use	Default value	Comment
`password`	The new better password	Required	None	

Request example:

http://localhost:8080/betty/better/UpdateBetterServlet?password=test

Better self-deregistration

Description	Removes the currently logged better from the server database, including all associated bets.
Since	0.90
HTTP request path	`/betty/better/RemoveBetterServlet`
Required security role	Better
HTTP request body	-
HTTP response	A HTTP response code only

Request example:

http://localhost:8080/betty/better/RemoveBetterServlet

Add or update a bet

Description	Submits a new bet, or updates an existing bet on an event's player. Such an action may only be performed until the event's date, at which time the event will be closed, and any further bet refused.
Since	0.90
HTTP request path	`/betty/better/BetServlet`
Required security role	Better
HTTP request body	-
HTTP response	A HTTP response code only

HTTP request parameters:

Name	Description	Use	Default value	Comment
`championship-id`	The championship identifier	Required	None	
`phase-id`	The championship's phase identifier	Required	None	
`event-id`	The championship's event identifier	Required	None	
`player-id`	The event's player identifier	Optional	None	Either this parameter must be specified, or the couple "`phase-ref`" / "`phase-rank`".
`phase-ref`	The dependency phase reference identifier	Optional	None	
`phase-rank`	The rank in the dependency phase	Optional	None	Integer value, must be strictly greater than 0
`result`	The bet result itself	Required	None	Integer value

Request example:

http://localhost:8080/betty/better/BetServlet?championship-id=formula1-2008t&phase-id=2008&event-id=AU&player-id=Heidfeld&result=1

Add or update a player's result

Description	Submits a new player's result, or updates an existing result in a championship's event. Such an action may only be performed after the event's date.
Since	0.90
HTTP request path	`/betty/admin/AddPlayerScoreServlet`
Required security role	Administrator
HTTP request body	-
HTTP response	A HTTP response code only

HTTP request parameters:

Name	Description	Use	Default value	Comment
`championship-id`	The championship identifier	Required	None	
`phase-id`	The championship's phase identifier	Required	None	
`event-id`	The championship's event identifier	Required	None	
`player-id`	The event's player identifier	Optional	None	Either this parameter must be specified, or the couple "`phase-ref`" / "`phase-rank`".
`phase-ref`	The dependency phase reference identifier	Optional	None	
`phase-rank`	The rank in the dependency phase	Optional	None	Integer value, must be strictly greater than 0
`result`	The bet result itself	Required	None	Integer value

Request example:

http://localhost:8080/betty/admin/AddPlayerScoreServlet?championship-id=formula1-2008t&phase-id=2008&event-id=AU&player-id=Heidfeld&result=2

Add or update a user

Description	Adds or updates a user (better and/or administrator) in the server database.
Since	0.90
HTTP request path	`/betty/admin/AddUserServlet`
Required security role	Administrator
HTTP request body	-
HTTP response	A HTTP response code only

HTTP request parameters:

Name	Description	Use	Default value	Comment
`id`	The user identifier (“login”)	Required	None	
`password`	The user password	Required	None	
`is-better`	The user has the better security role	Optional	Not set	Do not set this parameter to “`true`” or “`false`”: setting it always implies “`true`”, omitting it implies “`false`”. Can be combined with “`is-admin`”. At least one of these two security roles must be specified.
`is-admin`	The user has the administrator security role	Optional	Not set	Do not set this parameter to “`true`” or “`false`”: setting it always implies “`true`”, omitting it implies “`false`”. Can be combined with “`is-better`”. At least one of these two security roles must be specified.

Request example:

http://localhost:8080/betty/admin/AddUserServlet?id=test&password=test&is-better

Remove a user

Description	Removes an existing user from the server database, including all associated bets if any.
Since	0.90
HTTP request path	`/betty/admin/RemoveUserServlet`
Required security role	Administrator
HTTP request body	-
HTTP response	A HTTP response code only

HTTP request parameters:

Name	Description	Use	Default value	Comment
`id`	The user identifier (“login”)	Required	None	

Request example:

http://localhost:8080/betty/admin/RemoveUserServlet?id=test

Ingest a XML configuration file

Description	Adds or updates server database rows through an XML fragment. See for example "Add or update users", or "Specific championships". You may generally ingest multiple times the same XML configuration file, in particular when modifications have occurred. What you cannot do with this method is to **delete** existing elements (see the list of API described below concerning this topic).
Since	0.90
HTTP request path	`/betty/admin/AddXmlServlet`
Required security role	Administrator
HTTP request body	The XML contents
HTTP response	A HTTP response code only

Request example:

See the corresponding scripts in '`conf`' directory, '`addXML.bat`' and '`addXML.sh`'.

Remove a championship

Description	Removes a championship from the server database, including all associated phases and bets if any.
Since	0.90
HTTP request path	`/betty/admin/RemoveChampionshipServlet`
Required security role	Administrator
HTTP request body	-
HTTP response	A HTTP response code only

HTTP request parameters:

Name	Description	Use	Default value	Comment
`championship-id`	The championship identifier	Required	None	

Request example:

http://localhost:8080/betty/admin/RemoveChampionshipServlet?championship-id=formula1-2008

Remove a location

Description	Removes a location from the server database, including all child locations and associated events if any.
Since	0.90
HTTP request path	`/betty/admin/RemoveLocationServlet`
Required security role	Administrator
HTTP request body	-
HTTP response	A HTTP response code only

HTTP request parameters:

Name	Description	Use	Default value	Comment
`location-id`	The location identifier	Required	None	

Request example:

http://localhost:8080/betty/admin/RemoveLocationServlet?location-id=DZ

Remove a player

Description	Removes a player from the server database, including all associated event's player results if any.
Since	0.90
HTTP request path	`/betty/admin/RemovePlayerServlet`
Required security role	Administrator
HTTP request body	-
HTTP response	A HTTP response code only

HTTP request parameters:

Name	Description	Use	Default value	Comment
`player-id`	The player identifier	Required	None	

Request example:

http://localhost:8080/betty/admin/RemovePlayerServlet?player-id=Fisichella

Remove a resource

Description	Removes a resource definition from the server database, for each locale it defines, and for each championship if any.
Since	0.91, updated in 0.92
HTTP request path	`/betty/admin/RemoveResourceServlet`
Required security role	Administrator
HTTP request body	-
HTTP response	A HTTP response code only

HTTP request parameters:

Name	Description	Use	Default value	Comment
`resource-id`	The resource identifier	Required	None	

Request example:

http://localhost:8080/betty/admin/RemoveResourceServlet?resource-id=timezone

Reprocess scores, ranks and bets

Description	Re-processes all player event scores and phase ranks, and associated bets in a championship. See “Submitting event's results“.
Since	0.90
HTTP request path	`/betty/admin/ProcessChampionshipServlet`
Required security role	Administrator
HTTP request body	-
HTTP response	A HTTP response code only

HTTP request parameters:

Name	Description	Use	Default value	Comment
`championship-id`	The championship identifier	Required	None	

Request example:

http://localhost:8080/betty/admin/ProcessChampionshipServlet?championship-id=formula1-2008

Developer area

This section presents the developer's side of the “Betty” project : how to build it from scratch, and how it is organized.

Prerequisites

The following tools are not present in the source distribution. You have to download and install them all individually, either because of their size or because of licensing issues. All other libraries and tools needed to build Betty are already located in the source distribution (e.g. XDoclet 1, the Castor framework, etc).

Windows prerequisites

Sun JDK

Download the Sun Java SE Development Kit (JDK) 6 from the following Web page: http://java.sun.com/javase/downloads

Install the JDK for example in directory “`C:\Program Files\Java\jdk1.6.0_xx`”, where “`xx`” must be replaced by the appropriate update number (`12` at the time of writing).

Set the following system environment variables:

```
JAVA_HOME=C:\Program Files\Java\jdk1.6.0_xx
PATH=%PATH%;%JAVA_HOME%\bin
```

Apache Ant

Download Apache Ant 1.7.0 from the following Web page: http://ant.apache.org/bindownload.cgi

Install Ant for example in directory “`C:\apache-ant-1.7.0`”.

Set the following system environment variables:

```
ANT_HOME=C:\apache-ant-1.7.0
PATH=%PATH%;%ANT_HOME%\bin
```

Adobe Flex

Download the Adobe Flex 3 SDK from the following Web page: http://opensource.adobe.com/wiki/display/flexsdk/Download+Flex+3

Install the SDK for example in directory “`C:\flex_sdk_3`”.

Set the following system environment variable:

```
FLEX3_SDK_HOME=C:\flex_sdk_3
```

Open a command prompt and change directory to “`%FLEX3_SDK_HOME%\bin`”. Execute the following commands in order to build all needed specific Flex locales:

```
bin> copylocale en_US fr_FR
```

Google Web Toolkit

Download the Google Web Toolkit from the following Web page: http://code.google.com/webtoolkit/download.html

Install the kit for example in directory “`C:\gwt-windows-1.5.3`”.

Set the following system environment variable:

```
GWT_HOME=C:\gwt-windows-1.5.3
```

Linux prerequisites

Sun JDK

Download the Sun Java SE Development Kit (JDK) 6 from the following Web page: http://java.sun.com/javase/downloads

Install the JDK for example in directory "`/usr/java/jdk1.6.0_xx`", where "`xx`" must be replaced by the appropriate update number (`12` at the time of writing).

Set the following system environment variables:

```
export JAVA_HOME='/usr/java/jdk1.6.0_xx'
export PATH=$PATH:$JAVA_HOME/bin
```

Apache Ant

Download Apache Ant 1.7.0 from the following Web page: http://ant.apache.org/bindownload.cgi

Install Ant for example in directory "`/usr/java/apache-ant-1.7.0`".

Set the following system environment variables:

```
export ANT_HOME='/usr/java/apache-ant-1.7.0'
export PATH=$PATH:$ANT_HOME/bin
```

Adobe Flex

Download the Adobe Flex 3 SDK from the following Web page: http://opensource.adobe.com/wiki/display/flexsdk/Download+Flex+3

Install the SDK for example in directory "`/usr/java/flex_sdk_3`".

Set the following system environment variable:

```
export FLEX3_SDK_HOME='/usr/java/flex_sdk_3'
```

Open a terminal window and change directory to "`$FLEX3_SDK_HOME/bin`". Execute the following commands in order to build all needed specific Flex locales:

```
bob% copylocale en_US fr_FR
```

Google Web Toolkit

Download the Google Web Toolkit from the following Web page: http://code.google.com/webtoolkit/download.html

Install the kit for example in directory "`/usr/java/gwt-linux-1.5.3`".

Set the following system environment variable:

```
export GWT_HOME='/usr/java/gwt-linux-1.5.3'
```

Build

First of all, you have to download the JBoss Application server, version 4.2.3 (and optionally, the versions 4.2.2 or 4.2.1 if you target this specific version). Download it from the following Web page: http://labs.jboss.com/jbossas/downloads/

You don't need to uncompress these distribution packages. But you have to put the binary

distribution archive (namely "`jboss-4.2.3.GA.zip`", and optionally "`jboss-4.2.2.GA.zip`" or "`jboss-4.2.1.GA.zip`") in the directory one level up your Betty source directory. Let's say you uncompressed the Betty source archive in directory "`C:\Betty\Betty-1.1.0-src`"; in this case, you will place the JBoss archive in directory "`C:\Betty`" (for the Linux users, the same method applies).

This step is necessary if you want to test locally your server.

Windows build

Remember that all steps described in "Windows prerequisites" shall be performed first.

Open a command prompt and change to the root directory of the source distribution. Execute the following command in order to build the distribution:

```
Betty-1.1.0-src> ant
```

Linux build

Remember that all steps described in "Linux prerequisites" shall be performed first.

Open a shell terminal and change to the root directory of the source distribution. Execute the following command in order to build the distribution:

```
bob% sh ant
```

Ant targets

Here follows a list of the most useful targets in the main project build script ("`build.xml`"). For a more complete list, type "`ant -projecthelp`" at your command prompt (this Ant target will display the build system help).

Target name	Description
`all`	Builds in directory "`dist`" an image of the project binary distribution, combined with a full JBoss distribution. The Windows installation wizards and the distribution archives will just encapsulate this generated directory. When done, this target allows you to run the Betty server directly in directory "`dist`".
`clean`	Delete the generated directory "`dist`", and additional miscellaneous generated files. I often use it in order to be sure that my next build will be "unpolluted".
`mxmlc`	Builds the Flash client only. Useful when you just work on the client-side, and want to left the server intact. The good thing is that this target generates the Flash SWF file directly where the server (generated through the "dist" target) serves it.

If you plan to work on the Java (server) side, it's a good practice to generate the Java reference documentation of the project ("`ant javadoc`"). The current version of this documentation can be found at the following address: "http://betty.sourceforge.net/docs/javadoc/api/".

You may also generate the ActionScript reference documentation, if you plan to work on the Flex client ("`ant asdoc`"). The current version of this documentation can be found at the following address: "http://betty.sourceforge.net/docs/asdoc/".

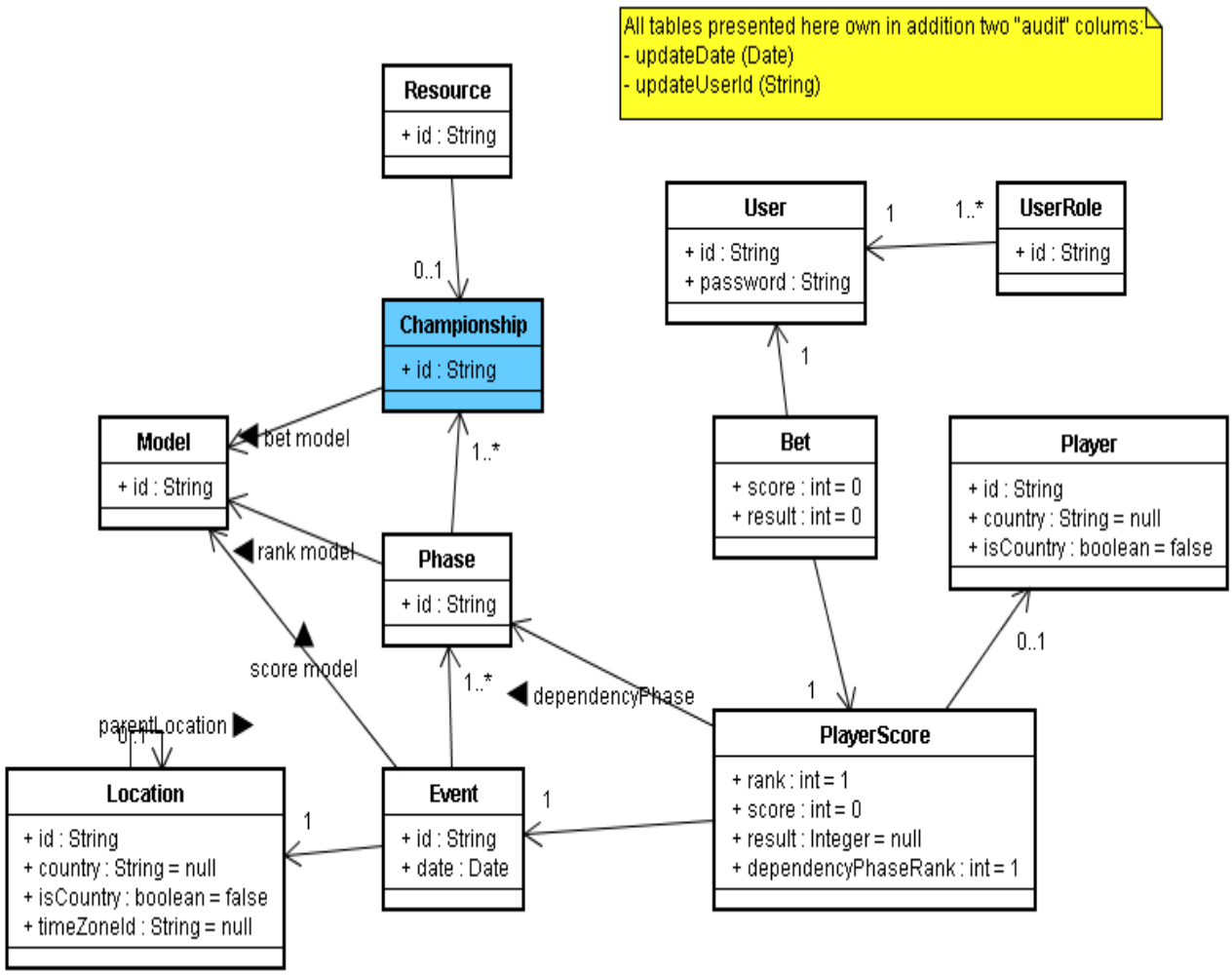
Source repository

The Betty source package is organized as follows:

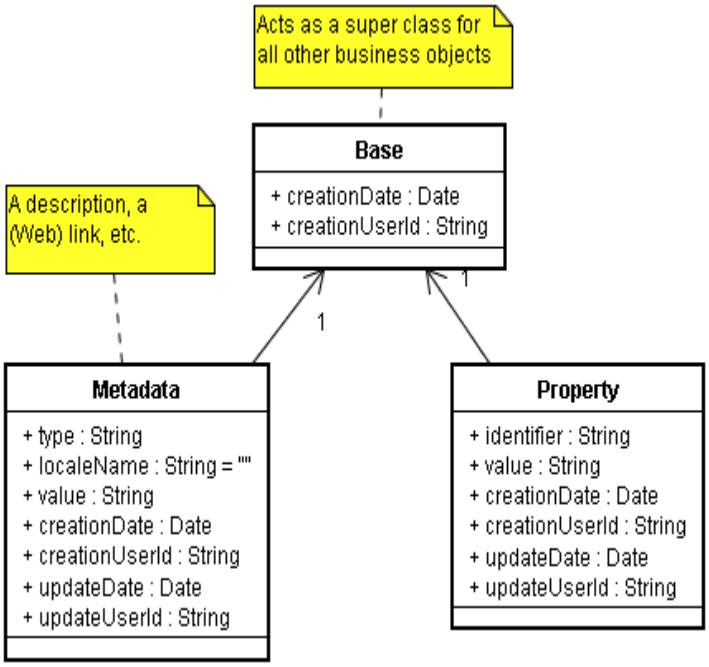
- Directory "**`conf`**": the XML configuration files of this project (see "Database preparation"), and associated ingestion scripts.
- Directory "**`docs`**": obviously all documentation-related files, including images, some licenses, the Jude project file, this file, etc.
  Other multi-purpose files used in particular for the documentation are located in the root source directory, namely "`LICENSE`", "`VERSION`", "`CHANGES`" and "`index.html`" (the main Web page of the project on the Internet).
- Directory "**`ext`**": all embedded third-party tools and libraries, including their license files.
- Directory "**`src`**": the project source files, separated per domain:
  - Java, GWT, Web pages and scripts for the Betty server ("`java`"),
  - Flex at the client side ("`flex`"),
  - installer-related projects ("`setup`").
- File "**`build.xml`**": the main Ant build script.
- File "`setup.iss`": The Inno Setup project used to build the Windows installation wizard.

Database structure

You will find the whole picture in the following schema:



In addition, almost all tables are associated to a "base" table, which really acts as a "super" table, from which we can hook general-purpose tables, like properties, metadata and so on:



Adding a new language

As a general rule, if the translation in the new language is the same as the English version, don't put it: the **default** language is always English.

And use the existing translations to understand as much as possible the meaning of each language resource.

In the Flash GUI

If you wish to add a new language in the Betty Flash GUI, the files to translate are described below, assuming that the new language is Italian for example (ISO 639-1 language code "`it`", Flash language code "`it_IT`"):

- « `conf/resources.xml` »: add the needed lines in each resource definition with the following format:
  `<description locale="it">Some text</description>`
  No source build is necessary to take your updates into account. You only need to re-ingest the file (refer to "Database preparation").
- « `src/flex/locale/it_IT/betty.properties.raw` »: you must create then fill in this file.

You may also translate some resources located in the following files:

- "`conf/locations.xml`" (search for the XML attribute"`locale`" for existing translations). You don't need to add translations for countries: they are automatically generated thanks to the underlying JDK Java class "`java.util.Locale`".
- The championship definitions in directory "`conf`".

When done, open a command window and change directory to "`%FLEX3_SDK_HOME%\bin`" or "`$FLEX3_SDK_HOME/bin`", according to your host OS. Execute the following command in order to build the new specific Flex locale:

```
copylocale en_US it_IT
```

Edit the file "`src/flex/Betty-config.xml`, and add the following line just under the XML

element “flex-config/compiler/locale”:

```
<locale-element>it_IT</locale-element>
```

Edit the main Ant build file, “build.xml”, find the target named “mxmlc”, and add the following block of instructions at the right place (you will find):

```
<antcall target="-iconv">
  <param name="dir" value="src/flex"/>
  <param name="lang" value="it_IT"/>
</antcall>
```

And finally build the Flash client ('ant mxmlc').

The list of languages currently supported by Flash but not translated in the GUI at the time of writing is the following:

- cs_CZ (Czech)
- da_DK (Danish)
- de_DE (German)
- es_ES (Spanish)
- fi_FI (Finnish)
- hu_HU (Hungarian)
- it_IT (Italian)
- ja_JP (Japanese)
- ko_KR (Korean)
- nl_NL (Dutch)
- no_NO (Norwegian)
- pl_PL (Polish)
- pt_PT (Portuguese)
- ru_RU (Russian)
- sv_SE (Swedish)
- tr_TR (Turkish)
- zh_CN (Simplified Chinese)
- zh_TW (Traditional Chinese)

In the Windows installation wizard

If you wish to add a new language in the Betty Windows installer, the files to translate are described below, assuming that the new language is Italian for example (language code “it”):

- “setup.iss”: the only section to look at in these “.INI” files is named « [CustomMessages] ». You only need to translate the first file, do some copy/paste on the second. Add the needed lines with the following format:
  it.xxx=Some text

Build the associated Windows wizard(s) to see your updates in effect (for example 'ant setup').

The list of languages currently supported by Inno Setup but not translated in the installation script at the time of writing is the following:

- `ca` (Catalan)
- `cs` (Czech)
- `da` (Danish)
- `de` (German)
- `es` (Spanish)
- `eu` (Basque)
- `fi` (Finnish)
- `he` (Hebrew)
- `hu` (Hungarian)
- `it` (Italian)
- `nl` (Dutch)
- `no` (Norwegian)
- `pl` (Polish)
- `pt` (Portuguese)
- `pt_br` (Brazilian Portuguese)
- `ru` (Russian)
- `sk` (Slovak)
- `sl` (Slovenian)