Weekly Progress Report

Group Name: Qracle

Erdem KARAHAN
Recorder, gatekeeper, optimist

This week I search through two new plug-ins “broadcast” and “packet filter” for openfire. When implementing the message listener plug-in I had problems about what classes and interfaces to use in the code; “packetfilter” plug-in was very useful to build up my task. Packetfilter plug-in is used for preventing some packets sent from some certain user and received by a certain user. The important point here to choose the type of the packet to capture the right packet, for example whether it is a presence packet or message packet. Yet I haven’t been able to accomplish my task, due to some problems. I’m planning to complete the message listener plug-in soon.

Taha Bekir EREN
Initiator, teamleader

This week I have implemented WebMethod calls from javascript. A WebMethod is either a web service method or a static method in a aspx codefile. These methods can be called from javascript asynchronously. Because of the request-response architecture of http, we will call some methods periodically from javascript to notice events such as a new message or a presence information. One important thing I noticed during the implementation is; when a webmethod is called from javascript, a new session is created which is different from the session of the web application. As I understand, with a call to a web method from javascript, a new web service session is created. This architecture causes a problem; when the webmethod is called how should it know the caller's context? To figure out the context of the caller we can pass the session ID or the user's id to the webmethod as a parameter. By this information, the webmethod will know which messages/events to process. Previously I mentioned the inefficient architecture of the ASP.NET Update Panel object. As an alternative I learned about an ASP.NET Ajax control, DynamicPopulate, which updates its inner html with an asynchronous call to a webmethod. This provides a fast and efficient update logic for our web application.
Next week I will continue my research in order to implement the web application prototype.

**Mine KARAKAYA**  
**Devil’s Advocate, Recorder**

I read more about updating process in ADO.NET; meaning submitting changes made in ADO.NET to my database in MSSQL. My topics were:

- how to fetch new autoincrement values generated by SQL Server after submitting updates to ADO.NET (by using batch queries, or by using rowUpdated property of DataAdapter object)

- UpdateRowSource property of Command object which takes an UpdateRowSource enumeration as argument to decide on fetching changed row’s new value from the database after the update process, or not.

- “SELECT @@IDENTITY” query to fetch last autoincrement value of last row added to database, which is generated by SQL.

- ContinueUpdateOnError property of DataAdapter object: by setting False for this property, in a condition where update process fails in any one of the rows, DataAdapter throws an exception; whereas by setting True for this property, even if a failure happens in update process, remaining changes take place in database.

Next week:

Sorting, searching, and filtering in ADO.NET