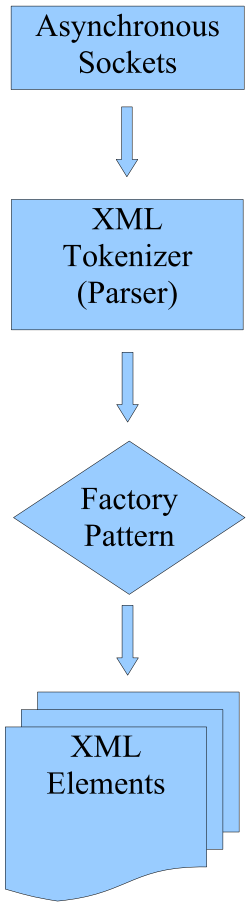
agsXMPP Software Design:



The aim of agsXMPP was to create a lighweight and extremly fast cross platform library for the XMPP protocol.

And we achieved this the following 3 step technology.

- Asynchronous sockets
- Fast XML Parser combined with factory pattern
- own light XML Dom which is the base for all agsXMPP protocol classes

Why did we reinvent the wheel and didnt use the Microsoft classes in the System.Xml namespace?

We decided to create our own small lightweight XML-Dom which is extremly fast, especially on embedded devices like PPC's and Smartphones.

The XmlTextReader was very good for SAX-like parsing. But Microsoft made changes in Service Pack 1 for .NET1.1 which doesn't allow us to parse network streams anymore. So we needed another XML parser.

Whats the magic of the core?

Once data is received from the socket, the data gets parsed from the sax-like xml parser. The parser is using a factory pattern for building elements of the associated agsXMPP protocol classes.

Example:

The socket receives a message and pushes the bytes to the parser. The xml parser detects a start tag named **message** which belongs to the **jabber:client** namespace. Before a element is created, the parser does a lookup in the factory hashtable. If a class is assigned in the table, a instance of the referenced type will be created. In this case it would create a new instance of the agsXMPP.protocol.client.Message class. If the table doesn't contain the name/namespace combination, it will create a instance of agsXMPP.Xml.Element.

All XMPP protocol classes are derived from agsXMPP.Xml.Element. They are 'abstract' elements that keep the xml tree in the memory. All properties are 'realtime properties'. When we want to read the Body of a message and call the Body property of the messaeg class, the class will looksup the <body/> Element at realtime.