A Chat Architecture

The **NWDatanode** class is a modified NWNReader, defined with a single underlying NAR, that not only reads and vaults but also uses **recordSlotEvents**() to read structured data inside text. When several different NWDatanodes are reading data from the same VAR tree, those nodes are combined into the **TopicTree** class. And these TopicTrees form the basis of the reading architecture. They are combined in an **NWChat** object that supervises their reading text.

NWChat

Topic TreeA

TopicTreeN

At the same time, for a chat application we need the concept of **data objects** and **metadata objects.** A metadata object has the responsibility of transferring info from one or more TopicTrees into a wrapped data object. It does this by implementing one or more **updateX**() methods that can be called by an owner:

Thus we have an application specific chat object like **NBChat** derived from NWChat and also containing metadata objects.

NBChat

TopicTreeA

TopicTreeB

metadataA

metadataB

yesno

Although the topic trees need not align exactly with the metadata, in practise they usually will be in a 1-1 relation. One kind of metadata is a **yesno** object, implemented locally within NBChat to handle pending confirmations.

Thus for incoming text, the NBChat will do the following:

1. Its underlying NWChat reads the text into all the Topic Trees. (Fills the vaults and slot event records)
2. The metadata updateX() methods are called (which transfers the vaults and records into the data sub objects).
3. The yesno metadata will process any pending confirmations
4. Choose a response that combines sub-responses from the metadata objects
5. Sets pending flag for any confirmations.