Why?

If you want to understand what I am doing here first read MoseCoreDump and know that I am convinced by what Mose says; if you want a core ***revolution*** look elsewhere (www.tikipro.org?). The whole point of this API is that it sits between (existing) functionality and the (existing) core, facilitating a core ***evolution***.

Advantages:

- New functionality is easier to write/integrate, particularly for those who are integrating software they know well (because they know what to chage in that code) but do not know Tikiwiki very well (because they don't have to go searching for the way to deal with things like users, databases, permissions etc.).
- New core functionality (eg LDAP authentication) is easier to write because you know what the requirements are - to serve the API functions. You don't have to worry about what everybody's code expects to be able to do with your data.
- It becomes possible to migrate the core incrementally without breaking functionality. Once you are sure that nothing outside the core accesses, for instance, users other than through API functions, you can completely refactor the core to improve performance, maintainability, compatibility etc. - or even to replace the core completely by integrating Tikiwiki into an existing system that provides those core functions.

Application Layers

The table below shows the relationship between elements of the system viewed as layers. Each layer is dependent only on the layer directly below it, therefore in order to manage change, only the interface between adjacent layers need be considered.

Presentation Layer
Logic Layer
Interface Layer
Core Layer

Templates, CSS					
Functionality					
API					
Users	Groups	Permissions	Request Authentication	Database Abstraction	Configuration

et

c.

..

Practical implications for me

When I am adding functionality or fixing bugs and I need to access some part of the core, I do it using an API function. If there is no function for what I want to do, I will write it. Some time later, I will document it. I will also probably take the trouble to document/edit documentation for any API functions written by others. Thus the API itself will evolve, in order to facilitate the evolution of the whole.

Practical implications for you (as a developer)

That's up to you. If you want to use API functions, go ahead - your code will be easier to maintain through future releases. If you want to write API functions go ahead (just comment with your SF id or other email so that the next point works). If you want to change someone else's API function ask them first, it's only polite.

Be aware that if you write buggy code and don't fix it before someone who is a fan of the API does, you might find your bugs fixed with API calls. If it upsets you, change it back but get it right!