Configuration Settings

MiServer stores its configuration settings in a number of XML files in the /Config/ folder.
MiServer uses a 2-tiered configuration system – one at the server level and one at the MiSite level.
The server level settings are comprehensive – every configuration setting is set.
MiSite level configuration settings override their server level counterparts.
A MiSite can run without any MiSite level configuration (it just uses the server level settings).

This gives you a few options:

1) Run with the server level settings – this is generally appropriate when you're running a single MiSite and you don't mind its settings possibly being overwritten when you update MiServer.
2) Run with mostly server level setting, overriding only those you care about at the MiSite level. This allows you to tailor your MiSite behavior, yet still pick up updates to server level setting that you haven't overridden.
3) Copy the server level configuration files to your MiSite. This allows you complete control to change whatever settings you care to, and protects you from settings being changed.

Note that with any of these options, new configuration settings introduced in a MiServer update will still be created at the server level when you update.

The Configuration Settings You Probably Care About

There are scores of configuration settings in MiServer, most of which you may never change. These will be comprehensively documented in the full MiServer User Guide. For this Quick Start Guide, here are the settings that you'll probably want to know about:

File	Setting	Significance to you
Server.xml	Port	The port which MiServer listens on. You'll probably want to change this to 80 if you a MiServer visible to the internet. You'll also want to change it if you care to run multiple instances of MiServer on the same machine.
Server.xml	ClassName	The class which implements core server functions. Normally, this is "MiServer". However, if you want to customize behavior on server startup or shutdown, or write an application server that interfaces with some business logic components, you'll want to change this. Note: Changing this setting means you'll need to write a class, derived from the MiServer class, that has the same name as the setting you provide.
Server.xml	Production	This is set to 0 by default thus enabling MiServer's debugging framework. This should be set to 1 for a production level server.
Server.xml	SessionTimeOut	This specifies the number of minutes to allow a client session to remain idle before it's considered timed out and closed.
Server.xml	TrapErrors	This is normally set to 0 to allow the developer to interactively debug errors. Set it to 1 on a production server and errors will be logged using the DrA error trapping framework.