Password

Why are plain text passwords in the config files?

Because there is no good way to "secure" them. When Tomcat needs to connect to a database, it needs the original password. While the password could be encoded, there still needs to be a mechanism to decode it. And since the source to Tomcat is freely available, the attacker would know the decoding method. So at best, the password is obscured - but not really protected. Please see the user and dev list archives for flame wars about this topic.

That said, any configuration file that does contain a password needs to be appropriately secured. That means **limiting access** to the file so that it could be read only by the user that Tomcat process runs as and root (or the administrator on Windows).

In The Cathedral and the Bazaar, Eric S. Raymond recounts a story where his fetchmail users asked for encrypted passwords in the .fetchmailrc file (which is almost identical to the situation posed here with server.xml). He refused, using the same arguments posed here: encrypting or otherwise obfuscating the password in server.xml does not provide any real security: only "security by obscurity" which isn't actually secure.

Auditors do not like this answer. In order to please auditors, feel free to do any of the following. Please be aware, that all of the following are "security by obscurity" and are not making Tomcat more secure. But it may allow you to pass an auditors checklist ....

- Use properties replacement so that in the xml config you have ${db.password} and in conf/catalina.properties you put the password there.
- Use `ServiceBindingPropertySource` to externalize your attribute values.
- Write your own `org.apache.tomcat.util.IntrospectionUtils.PropertySource` implementation to 'decrypt' passwords that are 'encrypted' in catalina.properties and referenced via ${...} in server.xml. You will need to set the system property `org.apache.tomcat.util.digester.PROPERTY_SOURCE` to point to your PropertySource implementation.
    - An example of a project that provides such custom PropertySource: Vault for Apache Tomcat.
- Since server.xml is an XML file — you can use XML entities. For example: "woot" becomes "woot" which is a way to obscure the password. You may even go through an extra layer of indirection by converting ${db.password} into XML entities so that the property replacement above is also performed. (But remember, while "clever", not more secure)
- XML entities can be read from an external file. That is, add the following text at the top of server.xml just after the XML declaration (`<?xml ...?>`) and before the `<Server>` element (line wraps can be removed):

  ```
  <!DOCTYPE Server [
    <!ENTITY resources SYSTEM "resources.txt">
  ]>
  ```

  Now, whenever you write `&resources;` in the text below, it will be replaced by the content of the file "resources.txt". The file path is relative to the conf directory.
- Write your own datasource implementation which wraps your datasource and obscure your brains out (XOR and ROT13 are great candidates for this since their strength matches the protection you'll actually get). See the docs on how to do this.
- Write your own `javax.naming.spi.ObjectFactory` implementation that creates and configures your datasource.

A cultural reference:

- It is turtles all the way down (Wikipedia)