Coding standards

Coding standards

Welcome to you, developper ! You have been elected committer on the project, or you want to contribute some code or some patch? This is great news. However, in order to be able to share your 'vision' and your code, some rules must be followed.

Hey, remember that those rules are not the best nor the worst, they are pretty much what they are for historical reasons, or for technical reasons, however, please, accept them as they are, and avoid religious war (please, oh please, no mail to say "WTF ? You are using spaces instead of tab ??? How stupid is this rule etc etc.) Rules are **always** stupid, but smart people follow them 😉

> ⚠ **eclipse IDE**
>
> Eclipse users can import those two files to enfore the code formating : formatting.xml and codetemplates.xml

> ⚠ **IDEA IDE**
>
> IDEA users can import this file to enfore the code formating : settings.jar

Headers

First, you **must** (and this rule accept no exception) use this header in top of all source file, or each file in which you can have comments :

```
/*
 *  Licensed to the Apache Software Foundation (ASF) under one
 *  or more contributor license agreements.  See the NOTICE file
 *  distributed with this work for additional information
 *  regarding copyright ownership.  The ASF licenses this file
 *  to you under the Apache License, Version 2.0 (the
 *  "License"); you may not use this file except in compliance
 *  with the License.  You may obtain a copy of the License at
 *
 *    http://www.apache.org/licenses/LICENSE-2.0
 *
 *  Unless required by applicable law or agreed to in writing,
 *  software distributed under the License is distributed on an
 *  "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
 *  KIND, either express or implied.  See the License for the
 *  specific language governing permissions and limitations
 *  under the License.
 *
 */
```

Class/Interface headers

Each **Class** or **Interface** should have an header which must contains :

- A descrption of this class/interface
- an **author** tag which should be :

```
@author <a href="mailto:dev@directory.apache.org">Apache Directory Project</a>
```

> *Thanks to avoid to put your name. The code is not yours, and much more important, but putting ypur name and e-mail, you will intimidate other developper ("Oh, no, I won't mess with this code, it has been developped by XXXX !") and second, you will receive mail in three years even if you have stopped all commitment on the project (and those who have sent you an e-mail will think that the project's memeber are not responsive...)*

If you use **html** tags, remember to escape '<' and '>' characters...

Static members and other members

Just add a sinle line javadoc comment like : **/*** blah ... ***/** before each member

Methods

Follow the standard **javadoc** rules : Description, **@param**, **@exception** and **@return**. It should be enough. No **@tags**, **@todo** tags, etc...

Escape **html** characters

Comments

No special rules, except that you should avoid :

- Useless comments like : i++; /* Increment i */
- Overusing comments : if you have to heavily comment a peice of code, then this piece of code might be too complex ...
- Speading little comments all over a method : if possible, write blocs of comments. The method header could hgenerally contains a full description of the code, and if it's not the case, just consider your method might be too long !
- Dead code commented. If it's dead, then put it in a cuffin. We use SVN, the Ressuscitator !

Basically, use your common sense 🙂

Naming

Naming ! Sounds like Blaming 🙂. Ok. We use **Sun(tm/c/r)** style :

- Constants are in UPPER CASE with accepted '_'
- Class starts with an uppercase and each starting word is upper cased. No '_', please !
- Methods starts with lower case and then follow the same rule than classes. No '_', please !
- Interfaces should not start with an 'I'
- Classes which implements an Interface must be followed by the postfix 'impl'
- Variables follow the method naming convention. No '_', please !
- Use meaningfull names.
- No double letter variables like ii, jj etc...

If you browse the code, you will see that many classes does not respect those rules. That's life ! Don't fix it if you don't touch a class. If you are fixing a method in a class, then you can change the code to respect the rules. Little by little, we may reach a stable state where all the code respect the rules 😉

Naming is really important for **APIs**. Be smart. If you are not sure, ask.

Spaces vs tabs

FOUR SPACES, NO TAB. **Final.**

No discussion. Using tabs break diffs. Modify your **IDE** to insert spaces when you use tabs, before it saves the file.

Formatting

Use the **formatting.xml** file which can be found in the **resources** directory in the root of the project. This is for **Eclipse**. If you don't use eclipse, then translate the formating to your favorite **IDE**

Use the **codetemplates.xml** file if you are using **Eclipse** too. You will find it at the same location. It brings you some standard headers for new classes, nex methods, etc.

Use **UTF-8** as a default for your files (except for properties, thanks to **java**, which should be in **ISO-8859-1**). Forget about exotic encoding...

DO NOT USE AN AUTOMATIC FORMATER FOR COMMENTS !!! People spend a lot of time making their comment looks like pretty, so if you just format them, you will have to recover the previous comments...

Some general rules :

- Always use '{' and '}' even for a single instruction, or if you have an empty block (don't use ';' for empty blocks)
- No more than one instruction on a single line, the only exception is the '?' ':' operation
- Use **this** to address the class variable if there is a risk of confusion (for instanc eif you have a parameter with the same name.
- Don't add a 'a_', or 'the_' before a parameter's name to distinguish it from the class variable which has the same name. Use **this** instead.

- Don't add **final** everywhere. Even if **final** is a substitute for **const**, it's semantic is not clear enough that you use it everywhere.
- Add spaces in method calls after '(' and before ')'
- '{' and '}' must be on the same column

This is a code example :

```
...
int result = myMethod( param1, param2 )

if ( result > 0 )
{
    // do something
}
...
```

Imports

Always declare all the classes you import, do not use **x.y.***

What else ?

Well, this was a very short introduction about coding rules. Use commen sense, look at what you see around you when adding some code, ask people about format, if you have a question.

That's it ! (I wait your comments, guys 🙂