Convention over configuration

Definition - What does Convention Over Configuration mean? Convention over configuration is a software development approach geared toward developing programs according to typical programming conventions, versus programmer defined configurations. It enables quick and simple software creation while maintaining base software requirements. Convention over configuration is also known as coding by convention. Convention over configuration relies on the development of a program through the use of an underlying language's native procedures, functions, classes and variables. This approach reduces or eliminates the need for additional software configuration files, ultimately facilitating and expediting software development, code consistency and maintenance. However, to follow these conventions, a software developer must be acquainted with the underlying framework. Software frameworks that support the convention over configuration development approach include Ruby on Rails, JavaBeans and CakePHP.

In the context of Grails, "coding by convention" means that a lot of (tedious and repetitive) explicit code and/or configuration is replaced by simple naming and directory structure conventions. For example: Any class whose name ends with Controller in the grails-app/controllers directory is automatically a Spring controller and closures defined in it will be bound to URLs - you don't have to configure this in an XML file as you would have to when using pure Spring. The same goes for taglibs (grails-app/taglib directory) - no more tedious TLD files! Domain classes in grails-app/domain probably have the most "convention magic", being automatically mapped to an automatically generated DB schema - with DB table and column names being by convention identical to the domain property names
Lots of conventions here: 1. How to name HTML elements so they're easily accessible as parameters from the HTTP request; 2. How to relate object attributes to table and column names in the database; 3. How to arrange a project into directories/packages; It's what you do when you find yourself solving a common problem in a particular style. You notice similarities and codify them into some automation scheme.
It means if you stick to certain coding conventions as defined by whatever convention-based framework you are using, you get a lot of functionality for free. In other words, if you structure your application in accordance to what the framework expects, a lot of work can be saved. It would be a good idea to look up the advantages and disadvantages of coding by convention.

Convention over configuration (also known as coding by convention) is a software design paradigm which seeks to decrease the number of decisions that developers need to make, gaining simplicity, but not necessarily losing flexibility.

The phrase essentially means a developer only needs to specify unconventional aspects of the application. For example, if there's a class Sale in the model, the corresponding table in the database is called "sales" by default. It is only if one deviates from this convention, such as calling the table "products_sold", that one needs to write code regarding these names.

When the convention implemented by the tool matches the desired behavior, it behaves as expected without having to write configuration files. Only when the desired behavior deviates from the implemented convention is explicit configuration required.

Convention in JAVA

What Is a Naming Convention?

A naming convention is a rule to follow as you decide what to name your identifiers (e.g. class, package, variable, method, etc..).

Why Use Naming Conventions?

Different Java programmers can have different styles and approaches to the way they program. By using standard Java naming conventions they make their code easier to read for themselves and for other programmers. Readability of Java code is important because it means less time is spent trying to figure out what the code does, leaving more time to fix or modify it.

To illustrate the point it's worth mentioning that most software companies will have a document that outlines the naming conventions they want their programmers to follow. A new programmer who becomes familiar with those rules will be able to understand code written by a programmer who might have left the company many years before hand.

Picking a Name for Your Identifier

When choosing a name for an identifier make sure it's meaningful. For instance, if your program deals with customer accounts then choose names that make sense to dealing with customers and their accounts (e.g., customerName, accountDetails). Don't worry about the length of the name. A longer name that sums up the identifier perfectly is preferable to a shorter name that might be quick to type but ambiguous.

A Few Words About Cases

Using the right letter case is the key to following a naming convention:

Lowercase is where all the letters in a word are written without any capitalization (e.g., while, if, mypackage).

Uppercase is where all the letters in a word are written in capitals. When there are more than two words in the name use underscores to separate them (e.g., MAX_HOURS, FIRST_DAY_OF_WEEK).

CamelCase (also known as Upper CamelCase) is where each new word begins with a capital letter (e.g., CamelCase, CustomerAccount, PlayingCard).

Mixed case (also known as Lower CamelCase) is the same as CamelCase except the first letter of the name is in lowercase (e.g., hasChildren, customerFirstName, customerLastName).

Standard Java Naming Conventions
The below list outlines the standard Java naming conventions for each identifier type:

Packages: Names should be in lowercase. With small projects that only have a few packages it's okay to just give them simple (but meaningful!) names:

```
package pokeranalyzer

package mycalculator
```

In software companies and large projects where the packages might be imported into other classes, the names will normally be subdivided. Typically this will start with the company domain before being split into layers or features:

```
package com.mycompany.utilities

package org.bobscompany.application.userinterface
```

Classes: Names should be in CamelCase. Try to use nouns because a class is normally representing something in the real world:

```
class Customer

class Account
```

Interfaces: Names should be in CamelCase. They tend to have a name that describes an operation that a class can do:

```
interface Comparable

interface Enumerable
```

Note that some programmers like to distinguish interfaces by beginning the name with an "I":

```
interface IComparable

interface IEnumerable
```

Methods: Names should be in mixed case. Use verbs to describe what the method does:

```
void calculateTax()

string getSurname()
```

Variables: Names should be in mixed case. The names should represent what the value of the variable represents:

```
string firstName
```

```
int orderNumber
```

Only use very short names when the variables are short lived, such as in for loops:

```
for (int i=0; i<20;i++)
{
  //i only lives in here
}
```

Constants: Names should be in uppercase.

```
static final int DEFAULT_WIDTH
static final int MAX_HEIGHT
```

Coding conventions are a set of guidelines for a specific programming language that recommend programming style, practices and methods for each aspect of a piece program written in this language. These conventions usually cover file organization, indentation, comments, declarations, statements, white space, naming conventions, programming practices, programming principles, programming rules of thumb, architectural best practices, etc. These are guidelines for software structural quality. Software programmers are highly recommended to follow these guidelines to help improve the readability of their source code and make software maintenance easier. Coding conventions are only applicable to the human maintainers and peer reviewers of a software project. Conventions may be formalized in a documented set of rules that an entire team or company follows, or may be as informal as the habitual coding practices of an individual. Coding conventions are not enforced by compilers. As a result, not following some or all of the rules has no impact on the executable programs created from the source code.

Naming Conventions IN Smarty:

Plugin files and functions must follow a very specific naming convention in order to be located by Smarty.

plugin files must be named as follows:

type.name.php

Where type is one of these plugin types:
function
modifier
block

compiler
prefilter
postfilter
outputfilter
resource
insert

And name should be a valid identifier; letters, numbers, and underscores only, see php variables.

Some examples: function.html_select_date.php, resource.db.php, modifier.spacify.php.

plugin functions inside the PHP files must be named as follows:

smarty_type, _name

The meanings of type and name are the same as above.

An example modifier name foo would be function smarty_modifier_foo().

Smarty will output appropriate error messages if the plugin file it needs is not found, or if the file or the plugin function are named improperly.

References:

http://www.smarty.net/docs/en/plugins.naming.conventions.tpl
http://en.wikipedia.org/wiki/Coding_conventions
http://ellislab.com/codeigniter/user-guide/general/styleguide.html
http://www.mediawiki.org/wiki/Manual:Coding_conventions/PHP
http://java.about.com/od/javasyntax/a/nameconventions.htm