Bubbly CSS

- Think of the task of filling a two-dimensional rectangular container with two-dimensional rectangular blocks. The container has 3 edges: top, left and right. The bottom is open and you can insert blocks through the bottom only. We call the container tab (as in browser tab).
- Think of the blocks to behave like helium balloons in a room, or bubbles in water. Blocks always bubble up to the top as far as possible. They do this with a special behavior: they bubble to the top along the right edge of the tab as far as possible and then move to the left as far as possible.
- Think of the blocks as being connected to each other like pearls on a pearl necklace. Blocks have a sequence.
- There are two types of blocks:
    - **container blocks** behave like tabs, i.e. they contain other pieces.
    - **atomic blocks**
- Tabs and blocks can be resized.
- If you resize the dimension of a container block, all contained blocks need to bubble in their positions again.

Let's go back to HTML reality for a minute. When you traverse an HTML page recursively, you get a document tree starting with the **html** root element. The document tree has multiple levels and at each level you can define a sort order of the elements based on the sequence in which they were inserted into the document. Elements on the same level normally have a predecessor and a successor except for the first one and the last one.

Back to the BWB. Let's assume that we have only one level of elements and we have the corresponding blocks lined up in their sequence at the bottom edge of the BWB. Let's make it even more simple. All of the lined-up blocks are inline blocks. Their size is fixed and you cannot shrink or grow them.

Now we start putting the blocks in one by one and inside the BWB by magic they fly to their final position. One by one, like bubbles bubbling up in a glass of water. There is a few things we need to know about inline blocks. They are not connected to each other, but they are eager to ensure that the sequence is preserved. They feel most comfortable when they are right in between their predecessor and their successor. Once you insert an inline block, by default, it bubble up towards what I referred to as the the top edge earlier and tries to squeeze right next to its predecessor. Sometimes there is not enough space next to the predecessor and then the inline block grumpily moves below and as far left as it can. You get the idea with inline blocks, they render word by word, line by line. There are some corner cases. If an inline block does not have a predecessor, it bubbles straight into the upper left corner of its parent.

Now we add container blocks to the lineup. Container blocks are posh folks. They don't want to mingle with other blocks, that's why they always start a new line when they bubble into their position and they always end with a line break, so the next block is for sure below. They grab the entire available width, and there is not much you can do about it. As I mentioned before, a container block is just a smaller version of the BWB, so it can contain elements that previously bubbled into their position inside the container block.

Now you get the idea of how blocks bubble into their final position when web pages are rendered. If you already know a bit more about CSS, you have already noticed that there are a lot of corner cases not covered yet. I will cover them in follow-up posts, so stay tuned.

To conclude this post, let's formalize what we have learned so far using proper CSS vocabulary.

Grails

Grails Domain Classes

Domain Class Creation

When you create a domain class, you can create them packaged or unpackaged. Just provide the (packaged) name as parameter (can be all small letters because Grails will take care of proper capitalization).

Attribute Types

When defining attributes, you need to type them so GORM can use the type for the object-relational mapping.

Constraints

When you define constraints you can use auto-complete to explore constraints options. All constraints are documented in the **Grails documentation => Quick Reference => Constraints**. These constraints are enforced in code by GORM's validation layer, but may affect the database scheme.

You can create you own validations by supplying a closure:

```
genre(validator: {val,obj -> //...})
```

When you are done with your domain class, you can create related controllers and views by right-clicking on the domain class and choosing **New => Generate Controllers and Views**.

You can rename a domain class and have dependent classed renamed automatically by Eclipse.

errors contains the names of the fields with validation errors. You can read the name of the first violated validation constraint.

Formulate constraints with () to be consistent with empty constraints for scaffolding sequence.

Check error properties in online documentation for testing constraints. Constraints should be chosen to be mutually exclusive for easier testing!

size and email means: value is optional, but if its there it needs to have certain shape. For non-optional add blank constraint.

toString() for localization

fields with constraint sized greater than 255 will be rendered as text area. Ordering constraints affects ordering of scaffold views fields.

Field Types for Domain Classes

Date

You can add special meaning to a `Date` filed by followoing this naming convention:

- `dateCreated`: set time stamp upon creation.
- `lastUpdated`: set time stamp upon save.

Many-to-One and One-to-one

When scaffolding these domain classes, you will first have to create an `ObjectB` before you can create an `ObjectA` since there is an implicit `nullable:false` constraint on `objectB`. This means that every `ObjectA` needs to reference an `ObjectB`, which is the meaning of **unidirectional** in this many-to-one relationship. An `ObjectB` can be owned by multiple `ObjectA`s. Deletion/save of `ObjectA` does not trigger deletion/save of `ObjectB`.

If you change the default constraint on `ObjectA` to

```
class ObjectA {
    ObjectB objectB

    static constraints = {
        objectB(nullable:true)
    }
}
```

you can compare and check that the scaffolding view makes `objectB` optional and of course the relationship turns into a normal many-to-one relationship.
You can turn the **unidirectional** many-to-one relationship and turn it into a **bidirectional** one-to-one relationship. `ObjectA` remains unchanged

```
class ObjectA {
    ObjectB objectB
}
```

and `ObjectB` changes to

```
class ObjectB {
    static belongsTo = [objectA:ObjectA]
}
```

ObjectA is still the owner of ObjectB, but ObjectB is now exclusively linked to its owner. Note that belongsTo turns objectB in ObjectA into a foreign key that cannot be null. Also note that there is an additional implicit nullable:false constraint on objectA. This will break the scaffolding view since to create either object, the other one needs to exist. The belongsTo configures a delete/save cascade from ObjectA to ObjectB.
In order to explore the belongsTo a bit more, you need to loosen the **bidirectional** character of the one-to-one relationship to be able to create objects in the view. Add nullable:true as constraint:

```
class ObjectB {
    static belongsTo = [objectA:ObjectA]

    static constraints = {
        objectA(nullable:true)
    }
}
```

Now you can create an ObjectB and then an ObjectA to which you assign the created ObjectB. This will instantly add ObjectA to ObjectB due to the belongsTo.

Note that with this configuration, you can create a nasty org.hibernate.HibernateException when you add the same ObjectB twice to an ObjectA. The view does not prevent you from doing it and only Hibernate notices that there is two ObjectAs that reference the same ObjectB. This you need to prevent via the view by adding a unique:true constraint for objectB:

```
class ObjectA {
    ObjectB objectB

    static constraints = {
        objectB(unique:true)
    }
}
```

Note that the foreign key is located in ObjectA. You can create the same one-to-one relationship with the foreign key located in ObjectB. We inverse the structure of the domain classes without changing the mutual ownership, i.e. ObjectA will still own ObjectB as before with the ownership defined through a cascading delete/save relationship:

```
class ObjectA {
   static hasOne = [objectB:ObjectB]
}
```

```
class ObjectB {
   ObjectA objectA
}
```

This results in the same situation as before, the view is broken because you cannot create ObjectA or ObjectB unless the other one exists. We cannot just add a nullable:true constraint on objectA in ObjectB since hasOne turns objectA of ObjectB into a foreign key that cannot be null. This is exactly opposite from as it was before.

To make the view work you need to add a constraint to ObjectA:

```
class ObjectA {
    static hasOne = [objectB:ObjectB]

    static constraints = {
        objectB(nullable:true)
    }
}
```

You can now test the cascading delete and will find when you delete an `ObjectA`, a linked `ObjectB` will be deleted, too. Just as before only with the foreign key in `ObjectB` instead of `ObjectA`. And not surprisingly, you can create another `org.hibernate.HibernateException` when you assing the same `ObjectA` to two different `ObjectB`s. Like before, this can be fixed by adding a `unique:true` constraint:

```
class ObjectB {
    ObjectA objectA


    static constraints = {
        objectA(unique:true)
    }
}
```

Make test A fix and add to two different Bs and B fix and add to two different A.

Python

In order to figure out which version of Python you are running, execute `python` from the command-line. This will yield the version of the Python interpreter.

Python Development

Package Installation

When you want to run a Python application for development or for production, you need to deploy it first. Every application comes with a `setup.py` that helps managing dependencies and get it deployed properly. Normally you download a distribution package, such as

```
package.tar.gz
```

and unzip it to a folder

```
package
```

The package will include a `setup.py`, which you need to execute from within `foo-1.0`:

```
python setup.py install
```

This will install the package into the active Python distribution.

When you develop a package (or an application), it is not feasible to copy your package into your active Python installation every time you want to run it. To this end, you can install a backage in **develop mode** by executing this command

```
python setup.py develop
```

This creates in folder `/sandbox/Lib/site-packages` a file `package.egg-link`, which contains the path to the folder where the source files

are located. Any changes made to the sources, will be immediately reflected in the Pythons sandbox without installing the package again.

Once you are done with development, you can remove the link using

```
pip uninstall package
```

PyDev for Eclipse

Python Interpreter Configuration

When developing Python with Eclipse, you need to create a virtual environment and install all required packages into the virtual environment.

1. In the global PyDev settings choose **PyDev => Interpreter - Python** and remove any non-sandbox interpreter definitions.
2. Click **New...** and choose the project name as interpreter name. There is one sandbox per project.
3. As interpreter executable choose `python.exe` in the `Scripts` folder of your sandbox.
4. Do not change any auto-selected elements with the following exception: check the `Lib` folder of the parent Python installation, e.g. `D:\Python27\Lib`. Note that this does **not** add the sub-folders of `Lib`, which is what we do not want to add `Lib\site-packages`, which contains custom packages of the parent Python installation.
5. Press **Apply** and **OK**.

Now you are ready to assign the Python interpreter to your project:

1. Right-click the project root and select **Properties**.
2. In setting **PyDev - Interpreter/Grammar** choose the Python interpreter that matches your project.

Pylons

Pylons Basics

Concepts

- Implements the Web Server Gateway Interface (WSGI) v1.0, known as PEP 333 (and meanwhile superseded by v1.0.1, known as PEP 3333).
- PEP 333 makes use of a `environ` dictionary which stores a number of CGI environment variables.
- In file `config/middleware.py` the application is assembled like an onion in function `make_app` and the entire application is returned.
- A typical Pylons application has the following layers:

```
Registry Manager
    Status Code Redirect
        Error Handler
            Cache Middleware
                Session Middleware
                    Routes Middleware
                        Pylons App (WSGI Application)
```

- It is the Routes Middleware that dissects the incoming URL and checks if it matches any URL configurations.

Project Structure

If you create **helloworld** project with

```
paster create -t pylons helloworld
```

you will have the following structure:

- `helloworld`
  - The runtime configuration `development.ini`
  - The application setup `setup.py`
- `helloworld/helloworld`
  - `config`

    - The basic Pylons environment variables are configured in `environment.py`
    - The configuration of the Pylons WSGI middleware is done in `middleware.py`
    - The URL configuration is done in `routing.py`
- `controllers`
- `model`
- `templates`
- `tests`

Running an Application

Running and loading an application is done with the `paster` command:

```
paster serve --reload development.ini
```

`--reload` ensures that the the server is automatically reloaded if any Python files or the `development.ini` are changed.

> In case you cannot terminate the server with `Ctrl + C`, you need to terminate all processes for `python.exe` and `paster.exe` in the task manager.

The `.ini` file contains a `[server]` block and an `[app]` block that define which application should be run on which server.

Pylons Controllers

> The virtual environment needs to be active and you need to execute the following steps from within the outer `helloworld` folder.

To create a controller, execute

```
paster controller hello
```

This creates

- `helloworld\helloworld\controllers\hello.py`
- `helloworld\helloworld\tests\functional\test_hello.py`

After you start with

```
paster serve --reload development.ini
```

you can access the controllers `index` method

Pylons Templates

When you create a Pylons project, you are asked which template handler to user an normally choose **mako**. When you choose mako upon project creation, you will find in your `base.py` an import

```
from pylons.templating import render_mako as render #@UnusedImport
```

that defines the `render` method.

Pylons with PyDev (Eclipse)

Shell Access

Working with Pylons, you will have to work quite a bit with the command-line. Therefore, you need to install the EasyShell from the Eclipse Marketplace in order to have easy access to the Windows Shell.

> There does not seem to be any good plug-in in Eclipse that supports using a shell from within Eclipse/PyDev. WickedShell works in principle but does not work properly when a virtual environment is active. So the best compromise seems to be getting easy access to the normal shell with EasyShell.

Installing Pylons into a Virtual Environment

Normally, in order to install Pylons with a PyDev project, all you need to do is install Pylons into a virtual environment with

```
pip install pylons
```

and then add that virtual environment as Python Interpreter to your PyDev project. However, currently there is a problem installing the following Pylons dependencies:

1. Paste
2. PasteDeploy
3. PastScript
4. repoze.lru

After automatically installing these packages into the virtual environment as dependencies, they are not recognized as Python packages in PyDev since for unknown reason the `__init__.py` is not copied. Without an init file, a folder structure is not recognized as package. You need to manually download the packages and copy the init file:

1. Copy `Paste-1.7.5.1\paste\__init__.py` to `sandbox\Lib\site-packages\paste`.
2. Copy `repoze.lru-0.5\repoze\__init__.py` to `sandbox\Lib\site-packages\repoze`.

Since PasteDeploy and PstScript are sub-packages of Paste, there is no need to copy their `__init__.py` from their `paste` folders since they are all identical.
Now all classes from these package should be recognized by PyDev and not throw an import not found error any more.

Creating the Pylons Project in Eclipse

There is no tight integration of Pylons into Eclipse. Therefore, a Pylons project is treated as a normal Python project. The first part of the Pylons setup is to create the Eclipse project which will contain the Pylons project **helloworld**:

1. Create a virtual environment **helloworld** with Pylons installed.
2. Add the **helloworld** virtual environment as Python interpreter with name **helloworld**.
3. Create a new PyDev project **helloworld**.
4. Point the Python interpreter settings to your sandbox.
5. Don't configure `PYTHONPATH` for the time being.

You have now an empty Python project with two files: `.project` and `.pydevproject`. Now you need to populate this PyDev project as Pylons project:

1. `cd` into the workspace folder of the PyDev project that you have just created.
2. Create the Pylons project with

   ```
   paster create -t pylons helloworld
   ```

3. Accept the default template engine and the default settings for `SQLAlchemy`.

You have now the Pylons project structure in your PyDev project. Note the nested **helloworld** folders. The outer folder is the PyDev project folder and the inner folder contains the actual source code of your Pylons project. You need to turn the inner **helloworld** folder into a `src` folder to add it to `PYTHONPATH`:

1. Open the project properties and select **PyDev - PYTHONPATH**.
2. Click **Add source folder** and select the *outer* **helloworld** folder.

Now you have the basic infrastructure in place.

Creating a Run Configuration

In order to be able to make use of the PyDev debugger for your Pylons project, you need to create a **Python Run** configuration:

Choose the `paster-script.py` of your Python virtual environment as main module, e.g.

```
D:\home\thilo\win-dev\sandboxes\helloworld\Scripts\paster-script.py
```

Choose as argument

```
serve development.ini
```

without the `--reload` option. With the `--reload` option, the server does not terminate properly when pressing the red button.

Leave all other settings unchanged.

Python for Scripting

If you want to write a custom script `command` in Python, simply create a file `command`. The first line in this file is a Python comment that tells the system where the interpreter lives, e.g.

```
#!/usr/local/bin/python
```

A better alternative is to indicate the location through an environment variable:

```
#!/usr/bin/env python
```

This makes your scripts transferable to other machines without having to change the location comment.

You can leave the file `comment` without the `.py` extension since you normally do not import a Python script into any other Python program.

Python Installation

Parent Installation

Python Installation

Normally, you create only one Python parent installation, i.e. either from the 2.7.x series or from the 3.x series. Pylons currently requires Python 2.7.x.

1. Download the latest Python Windows installer, e.g. `python-2.7.3.msi` or install via your package manager on Linux.
2. Install for all users and choose `D:\Python27` as installation directory in Windows.
3. Don't customize, install everything.
4. Go to **Start => Settings => Control Panel => System => Advanced => Environment Variables** and verify that your installation directory `D:\Python27` is part of `PATH`.
5. Type `python` at the command-line to verify that it is setup properly.

Distribute Installation

`distribute` is a Python package to easily download, build, install, upgrade, and uninstall Python packages.

1. From http://python-distribute.org/ download `distribute_setup.py`.
2. Execute `python distribute_setup.py`.
3. Make sure that `D:\Python27\Scripts` is part of `PATH`.
4. Verify your installation by typing `easy_install`.

`pip` Installation

`pip` is the `easy_install` replacement and ironically needs to be installed with `easy_install`, which is available from the `distribute` install ation. Type

```
easy_install pip
```

Type `pip` to verify that it works. You can complement `pip` with `yolk` which helps with package management:

```
pip install yolk
```

`virtualenv` Installation

In order to use virtual environments, install package `virtualenv`:

```
pip install virtualenv
```

For any projects, both development and production, you make use of virtual environments. Any project-specific packages need to be installed into a project-specific sandbox.

Python Package Management

The old approach to package management for Python is using `easy_install` as part of the `setuptools`. This approach is deprecated and replaced with the combination of `pip` replacing `easy_install` and `distribute` replacing the `setuptools`. Both approaches help installing packages from the Python Package Index (PyPI).

Make sure you complete the parent installation before proceeding.

Before Installing a Package

Check the package information at

```
http://pypi.python.org/pypi/<package>/
```

Check which version will be installed using

```
yolk -V <package>
```

Check which version are available at

```
http://pypi.python.org/simple/<package>/
```

Check dependencies by downloading the `.tar.gz` or `.egg` and search for `install_requires` inside the package's `setup.py`.

Package Management with `pip`

`pip` is a replacement of `easy_install` for Python package management. `pip` is by default included in every virtual environment. `pip` is the choice over `easy_install`.

Install

Install a Python package with this command:

```
pip install package_name
```

This installs package *—package_name* from PyPI, along with its dependencies. You can also install a specific version of the package:

```
pip install package_name==1.0.4.
```

The `pip install` command will install the package in in folder `../Lib/site-packages` (of your virtual environment).

Upgrade

To upgrade an installed package to the latest version with

```
pip install --upgrade package_name
```

or upgrade to a specific version using

```
pip install --upgrade package_name==1.04
```

Uninstall

You can uninstall a package using

```
pip uninstall package_name
```

Yolk

`yolk` helps you figure out which packages and versions you have installed and which versions of packages are available to be installed with `pip`. Install `yolk` with `pip` and use it as follows:

Command	Description
`yolk -l`	List all installed packages
`yolk -U`	Check for which of the installed packages updates are availalbe
`yolk -V`	Check which versions of a package are available on the PyPI.

When you work with an active virtual environment and have no `yolk` installed in that environment, the global `yolk` will be executed and applied to the global Python installation. Therefore make sure that you install `yolk` into every virtual environment.

Requirements File

Normally your Python project depends on many packages, each of which must be installed in a specific version. This set of packages is called an environment. A requirements file helps to define an environment.It contains a list of packages an some version information. The requirements file is saved as `requirements.txt` and you can install the environment with

```
pip install -r requirements.txt
```

To create a new requirements file from an environment that is known to work, use

```
pip freeze > requirements.txt
```

This will write all installed packages and their precise versions into the file. Using a requirements file is the method preferred over installing a bundle. For each and every Python project you should maintain a requirements file and thereby precisely define which versions of packages are to be used.

Legacy Package Mangement with `easy_install`

While `pip` is a must in virtual environments, it is not readily available in the parent Python installation on Windows. As explained for the parent installation, you need to manage package `virtualenv` in the parent installation with `easy_install`. This requires the following basic commands:

To **install the latest version** of a package, use

```
easy_install <package_name>
```

To **install a specific version** of a package, use

```
easy_install "<package_name>==2.0"
```

To **upgrade to the latest version**, use

```
easy_install --upgrade <package_name>
```

To **uninstall** a package, use

```
easy_install -m <package_name>
```

This will only unregister the package, but not remove the actual installation files. You can remove them manually after executing this command.

Python Eggs

Eggs are to Pythons as Jars are to Java.

Python eggs (`.egg` zipfiles) are a way of bundling additional information with a Python project, that allows the project's dependencies to be checked and satisfied at runtime, as well as allowing projects to provide plugins for other projects. They are basically additional folders added to `PYTHONPATH`. The main advantages of Python eggs are:

- Can be easily installed with `pip`.
- They are zero installation, i.e. no build or install step is required. Just put them on `PYTHONPATH`.
- They can include meta-data about dependencies.
- They can be used as plug-ins.
- You can build an egg from a packages `setup.py` using `setuptools`.

Virtual Environments

Using virtual environments requires the parent Python installation to be completed.

Virtual Environment Location

You need to create a virtual environment on a per project basis. You create for each Python project a separate virtual environment in

```
D:\home\thilo\win-dev\sandboxes
```

using the same folder name as the original project folder. You can optionally back up this folder into the Wuala `win-dev` folder.

The normal restore procedure for sandboxes is to create a new sandbox and then install required packages with a requirements file.

Virtual Environment Creation

To create the sandbox inside a folder `sample_project`, change to the parent directory and execute

```
virtualenv <sandbox>
```

Here is the fine print:

- By default `virtualenv` ensures that no packages that are already installed into the parent Python installation are shared with the virtual environment. This is to properly separate parent environment and virtual environment.
- Do not put a sandbox folder under version control. Instead create a requirements file and version control it.
- The magic of the sandbox is inside the `Scripts` sub-folder.
- The `Scripts` folder contains `easy_install` and `pip`. Use `pip` only.

Virtual Environment Activation and Deactivation

When you want to use the virtual environment within Eclipse, you do not need to activate it. Just point Eclipse to the sandbox folder.

If you want to make changes to the virtual environment, you need to activate it by invoking a script inside the `Scripts` folder:

- `activate.bat` to activate the virtual environment:

```
D:\home\thilo\win-dev\sandboxes\sandbox\Scripts\activate.bat
```

- `deactivate.bat` to deactivate the virtual environment:

```
D:\home\thilo\win-dev\sandboxes\sandbox\Scripts\deactivate.bat
```

This will ensure that when you use the `pip`, `easy_install`, or `setup.py` commands, it will operate inside the sandbox and not pollute the parent Python installation.

In order to check the shell path of your active virtual environment, type `python` to invoke a Python console and type

```
import sys
sys.path
```

Tomcat

Directory Structure

Directory	Description
`CATALINA_HOME`	Installation directory:
`/bin`	Contains the startup and shutdown scripts.
`/conf`	Contains the main configuration files including `server.xml` and `web.xml`.
`/lib`	`.jar` files that are shared across all Tomcat components. All web applications deployed to Tomcat can access these libraries. Libraries include Servlet API and JSP API.
`/logs`	Contains Tomcat's log files.
`/temp`	Temporary file system storage.
`/webapps`	The directory where all the web applications are deployed and where you place your `.war` file when it is ready for deployment.
`/work`	Tomcat's working directory where it places all servlets that it generates from JSPs.

Catalina Command-line arguments

The Catalina script is the main script to control Tomcat:

```
MP4Box -add video.h264 -add audio.ac3 output.mp4
```

⚠ Any MP4 files are temporary and will not be archived!

Other Video-Related Tasks

- Any type of video processing can be done done with open source editor Open Shot.
- You can outsource video encoding to HeyWatch.
- Video conferencing can be done with ViVu.
- Minitube is a client for downloading YouTube videos in Ubuntu.

Keyboard Shortcuts

Adobe Lightroom

Shortcut	Action
`Command + A`	Select all
`Command + D`	Deselect all

Google Chrome

Developer Tools

Shortcut (PC)	Shortcut (Mac)	Action
	`Alt + Cmd + U`	View source
`Ctrl + Shift + I`	`Alt + Cmd + I`	Open DevTools
`Ctrl + Shift + J`	`Alt + Cmd + J`	Open DevTools and bring focus to console
`Ctrl + Shift + C`	`Alt + Cmd + C`	Toggle inspect element mode (equivalent to clicking on magnifying glass)
	`Cmd + E`	Start/stop recording in Timeline panel
	`Cmd + S`	Save
`Esc`	`Esc`	Toggle console

Browser

Shortcut (PC)	Shortcut (Mac)	Function
`F5`	`Cmd + R`	Refresh
`Shift + F5`	`Shift + Cmd + R`	Force reload (refresh without caching)
`Ctrl + Shift + N`	`Cmd + Shift + N`	Open tab in incognito mode
`Ctrl + 1`	`Cmd + 1`	Select leftmost tab
`Ctrl + 2` to `Ctrl + 8`	`Cmd + 2` to `Cmd + 8`	Select each of the tabs from left to right
`Ctrl + 9`	`Cmd + 9`	Select rightmost tab
`Ctrl + W`	`Cmd + W`	Close active tab
`Ctrl + T`	`Cmd + T`	Open new tab

`Ctrl + Tab`	`Ctrl + Tab`	Select next tab (left to right)
`Ctrl + Shift + Tab`	`Ctrl + Shift + Tab`	Select previous tab (right to left)
`Ctrl + Shift + T`	`Cmd + Shift + T`	Undo close tab

Extensions

- Chrome Sniffer
- Contactually
- Flattr
- Ghostery
- HTTPS Everywhere
- LastPass
- Pocket
- Stay Focused
- Web Developer
- Yet another flags

IntelliJ IDEA

You can find the latest default keymap here.

Code Completion

Shortcut	Action	Comment
`Tab` or `Space`	Insert suggested element	When typing, IntelliJ makes suggestion for code completion. Insert element that is at the top of the list.
`Down` then `Enter`	Insert selected element	Navigate to element of choice for typing code completion.
`Ctrl + Shift + Enter`	Insert selected element	And afterwards clean up code, and go to new line.
`Ctrl + Enter`	Insert first element (without focus)	
`!` or `.` or `,` or `;`	Insert selected element	Exclamation mark: insert and negate. Dot: insert and set dot. Comma: insert and set comma. Semicolon: insert and set semicolon.
n/a	Middle matching	No need to type from beginning. Type anything from the middle or type camel prefixes.
`Ctrl + Space`	Basic code completion	When you need assistance before typing anything. Shows only classes that have been imported. In order to get access to classes that are not imported yet, press `Ctrl + Space` and do a second basic code completion (access to accessible classes).
`Ctrl + Shift + Space`	Smart code completion	Narrow down suggestion list to what is expected in context.
`Ctrl + P`	Parameter info	Execute from within method call arguments.

Code Exploration

Shortcut	Action	Comment
`Alt + F7`	Find usages	Find usages in project files.
`Ctrl + F7`	Find usages in file	

Ctrl + Shift + F7	Highlight usages in file	
Ctrl + Alt + F7	Show usages	Filter by read/write access.
Ctrl + F12	File structure popup	
Ctrl + Shift + I	Open quick definition lookup	Quick review of the definition of the element at cursor.
Ctrl + B or Ctrl + Click	Go to declaration	Go to declaration of element at cursor.
Ctrl + Q	Quick documentation lookup	
Ctrl + H	Type hierarchy	

Line Editing

Shortcut	Action	Comment
Ctrl + Y	Delete line at caret	
Ctrl + Shift + J	Smart line join	Join adjacent lines. Place cursor anywhere in upper line.
Ctrl + Enter	Smart line split	

Navigation

Shortcut	Action	Comment
Alt + Home	Show navigation bar	
Alt + Right	Go to next editor tab	
Alt + Left		
Ctrl + N	Go to class	
Ctrl + Shift + N	Go to file	
Ctrl + Alt + Shift + N	Go to symbol	
Ctrl + G	Go to line	
F12	Go back to previous tool window	
Esc	Go to editor (from tool window)	
Shift + Esc	Hide active or last active window	

Task Management

Shortcut	Action	Comment
Alt + Shift + N	Create task	
Alt + Shift + T	Switch to task	

Mac OS X

Text Navigation

Shortcut	Action
Command + Left	Move to beginning of line
Command + Right	Move to end of line
Alt + Left	Move to beginning of current/next word
Alt + Right	Move to end of current/next word
Command + Up	Move to beginning of all text
Command + Down	Move to end of all text

Text Selection

Shortcut	Action
Shift + Command + Left	Select text to beginning of line
Shift + Command + Right	Select text to end of line
Shift + Alt + Left	Select text to beginning of current/next word
Shift + Alt + Right	Select text to end of current/next word
Shift + Command + Up	Select text to beginning of all text
Shift + Command + Down	Select text to end of all text

Shell

Shortcut	Function
`Arrow up`, `Arrow down`	Scroll through recently used commands
`Home`	Move cursor to beginning of line
`End`	Move cursor to end of line
`Alt + B`	Move cursor back (one word)
`Alt + F`	Move cursor ahead (one word)
`Alt + D`	Delete word
`Ctrl + K`	Delete until end of line
`Ctrl + T`	Swap the two preceding characters
`Alt + T`	Swap the preceding words
`Tab`	Expansion
`Ctrl + L`	Clear screen
`Ctrl + R`	Search for previously used command

Trackpad and Magic Mouse

Task	Trackpad	Magic Mouse	Keyboard Shortcut

Tap to click	Tap with one finger		
Secondary click	Click on right side	`Ctrl` + Click	
Look up	Tap with three fingers		
Three finger drag	Move with three fingers		
Scroll direction	Natural	Natural	
Zoom in or out	Pinch with two fingers		
Smart zoom	Double-tap with two fingers	Double-tap with one finger	
Rotate	Rotate with two fingers		
Swipe between pages	Scroll left or right with two fingers	Scroll left or right with one finger	
Swipe between full-screen apps	Swipe left or right with four fingers	Swipe left or right with two fingers	
Notification center	Swipe left from the right edge with two fingers		
Mission Control	Swipe up with four fingers	Double-tap with two fingers	`Ctrl + Up`
App Exposé	Swipe down with four fingers		`Ctrl + Down`
Launchpad	Pinch with thumb and three fingers		
Show Desktop	Spread with thumb and three fingers		

Ubuntu Unity

Launcher

Shortcut	Function
`Super` (hold)	Show launcher
`Super` (hold) then `1`, `2`, ..., `0`	Put focus on selected application
`Super` (hold) then `Shift` then `1`, `2`, ..., `0`	Open new instance of selected application if it is already open
`Super` + `T`	Open trash
`Ctrl` + `Alt` + `T`	Launch terminal window
`Alt` + `F1`	Show launcher and enable arrow key navigation

Dash

Shortcut	Function
`Super`	Open dash
`Super` + `A`	Open dash (applications lens)
`Super + F`	Open dash (files and folders lens)
`Shift` + `Tab` (when dash is open)	Move to previous lense
`Tab` (when dash is open)	Move to next lense
`Alt` + `F2`	Open dash in command mode

Panel

Shortcut	Function
F10	Open first menu on panel.
Esc	Close menu without choosing anything.

Window Management

Shortcut	Function
Super + W	Spread mode across all workspaces.
Alt + Shift + Up	Spread in current workspace.
Ctrl + Alt + D	Show desktop (minimize or restore all windows).
Alt + Tab	Switch to other active windows. Active windows are grouped by application. There is marker in icon when an application has more than one window. Let go of Tab and continue pressing Alt to switch between windows of selected application.
Alt + F4	Close active window.
Alt + F7 then arrow keys	Move active window.
Alt + F9	Minimize active window.
Alt + F10	Expand active window to full screen.

Workspace Management

Shortcut	Function
Super + S	Zoom out on all workspaces.
Ctrl + Alt + Arrow	Switch between workspaces (arranged in a square).
Ctrl + Alt + Shift + Arrow	Move active window to another workspace.
Ctrl + Alt + L	Lock screen.
Alt + PrtSc	Take screenshot of current workspace.

vim

Navigating

Shortcut	Function
w	jump by start of words (punctuation considered words)

Editing

Shortcut	Function
dd	delete (cut) current line
dw	delete (cut) current word
i	start insert mode at cursor
I	Insert at the beginning of the line
s	delete character at cursor and substitute text

Saving and Exiting

Shortcut	Function
`:w`	write (save) without exit
`:wq`	write (save) and quit
`:q`	quit (works only if there are no unsaved changes)
`:q!`	quit and throw away changes

Systems Management

Apple OS X

Change Host Name

1. Invoke **System Preferences => Internet & Wireless => Sharing**.
2. The first line displays the current host name.
3. Click **Edit** and change.

Common Directories

Directory	Description	Partitioning Candidate
`/home`	Personal files for user accounts. If you maintain `/home` as a separate partition, you can install new versions of a distribution or even different distributions altogether without wiping out any user settings.	yes
`/opt`	A lot of third-party software is known to install under `/opt`. On some systems the overall root partition may need to be only a few gigabytes with the bulk of the growth in `/opt` .	yes
`/tmp`	Many programs use `/tmp` to store temporary files that don't have to persist after a reboot. A common problem, though, is that a program might store far too much data in `/tmp`, and if it is part of the root partition, the entire root partition can fill up.	yes
`/usr`	The `/usr` directory should change only after you install new packages, upgrade the system or single programs are updated. Generally speaking, it is set up so that it changes only when you upgrade programs or install new packages. Because of that, some security-minded administrators put it as its own partition and mount it read-only during normal operation to prevent attackers from replacing programs with Trojan horses or other viruses.	yes
`/usr/local/bin`	Part of `PATH`. Create symbolic links to executable files of custom software from within this directory.	no
`/usr/local/man`	Create symbolic links to man pages of user-added software from within this directory.	no

`/var`	Designed for storing data that varies in size. From mail spools to HTML files to system logs, this directory can often grow pretty quickly. Separate `/var` out to its own partition so that if anything grows rapidly and fills up the partition, it is much easier to recover from than if the entire root partition were to be full.	yes
`/boot`	In the past there have been different limitations placed on Linux boot loaders. A common workaround, however, has been to create a small (64–128MB should be enough) partition at the beginning of the disk for the `/boot` directory.	yes

Synchronize Google Calendar

1. Go to **System Settings => Internet & Wireless => Mail, Contacts & Calendars**.
2. Choose account type **Gmail**.
3. Enter Email and account-specific password.
4. Choose Calendar only and click on **Add account**.

Synchronize Google Contacts

1. Open **Contacts => Preferences => Accounts**.
2. Click **Synchronize with Google**.
3. Enter Email and application-specific password.

DNS

Whois

Simply execute

```
whois <domain_name>
```

to query whois registration data. Whois figures out which whois server to query. Normally it is `whois.nic.<TLD>`. If you want to figure out the authoritative whois server, go to IANA's whois for TLDs, https://www.iana.org/whois. Query a TLD and check for whois record. Authoritative query:

```
whois -h <whois_server> <domain_name>
```

United Domains

You can request a **Zonenauszug** from United Domains.

maier.asia

Dates

Created	2008-02-22
Next Extension	2013-07-21

DNS Server

Primary DNS Server	`ns.udagdns.de`
Secondary DNS Server	`ns.udagdns.net`