This document assumes that the environment has already been set up. That is to say, your IDE has been configured to work with spring, and the necessary gradle/maven shit has been sorted out.

Initialization

**ApplicationContext**

ApplicationContext is a class and is one of the first things you need to instance in your main.

This thing is used to specify the Framework API used for creating and controlling instances of objects throughout the project. In the tutorial I’m following, they set it to be a new ClassPathXmlApplicationContext(“Beans.xml”). Which just means that the framework stores objects in a bean configuration file, which is conveniently in xml format.

**Bean configuration in xml – AKA why Lauritz is going to hate spring**

To start, you are going to have to make a configuration file. This file should be placed in the src folder of your project. The name of the file is irrelevant, but the meta naming of it is Beans.xml.

*Dafuq’s a bean anyway?*

So, a bean is just a class which has been given an id (in the form of a string) in the bean config file. In the bean config file there is a tag called:

Inside this tag, you can insert tags which registers classes as beans by using the following syntax:

It should be noted that the class path here is referring to the package path leading to the class. So say I have a package called main containing another package called project, which contains the class ‘DankSlap’. The class path in the above syntax would then be replaced with “main.project.DankSlap”. Remember the quotes as that is also in string format. The id is just a string you use to refer to the class with throughout the code.

Now, within the bean tag, you can then add properties like so:

**Something something containers.**

So you might have heard some buzzwords about spring working with containers, and were wondering dafuq it was. Well, remember that context class at the beginning? That’s the container. It’s just an object which takes meta data from the config and POJO objects to produce results.

**POJO?**

Google it. It takes like a minute to understand with the Wikipedia article on it, yet it forms the basis of how to use spring.

Beans nom nom

**More detailed look on those beans**

Remember how you could specify id and class for a bean? Well, turns out there’s even more stuff to be specified. I’m just gonna list them below:

* Name
  + Alternative to id. Not sure of the differences. Both are used to identify which bean is which. I think you might be able to create multiple beans for the same class, so long that they have different ids/names.
* Scope
  + Defines some things about how the bean is used. There are 5 different string parameters it can be given:
    - singleton
      * Beans with this scope have 1 persistent instance of an object. That means no matter how many times that bean is used, it will always return the same object. This also means that if you assign an object to a bean, the bean will not be able to have its object changed later. Attempts to do this anyway will result in any calls made on the second object to be made on the first object instead.
    - prototype
      * Unlike the singleton, allows the bean to have changing objects. This means that calls directed to object 1 are always made on object 1, and calls directed to object 2 are always made on object 2, etc.
    - request
    - session
    - global-session
  + request, session, and global sessions refers to http.
* constructor-arg
  + Dependency injection
* properties
  + Dependency injection
* autowiring mode
  + Dependency injection
* lazy-initialization mode
  + Funny lil’ Boolean. If true, the bean will only be instanced on request rather than on application start-up.
* initialization method
  + Specifies a method in the object to be run after the other bean properties have been set. Said method has to be of void datatype and take exactly 0 parameters. If you want to be ass-backwards, you could instead make the class in question implement InitializingBean and make a method called *public void afterPropertiesSet()* exactly.
* destruction method
  + Same as above, except it runs the method at the end of the bean’s lifetime.

*Sidenote:*

Have a ton of beans with inits and destroyers using the same name? Are you tired of having to ensure everything is named the same to a tee? Look no further! Remember the *[SOMESHIT]* attributes? Turns out, you can add the attributes *[default-init-method = “String]* and *[default-destroy-method = “String”]* to specify inits and destroyers for **all** of the beans.

Web MVC Framework

Also known as the web model-view-controller framework.Not quite sure if it’s necessary for web development, or if it’s just some addon with extra features to compliment standard spring.