Spring Crib Sheet

Jonathan Gibbons
Tall Software Limited

All you need to know! (for the average user, on green field project)
Source: C:\<install dir>\spring-framework-1.1.3\docs\reference\html_single\index.html

Note from the author.
I only use Spring for IOC, not for AOP, not for DAO, not for JDBC rewrite, not for Web, or every other part of my system. I have a JVM and an OS, I do not need yet another J2EE. This is my personal choice, and one I am extremely happy with.

All the notes you need on Spring IOC.

- Avoid anything that ties your code to the spring framework.
- Do use **ApplicationListener** to get refresh and close events for your singletons on close down (even though it is a spring interface).
- Use setters and not constructors for injection.
- For most users, beans in the container will be singletons. By default all beans are singletons. They are created by the app context at start-up unless **lazy-load="true"**;
- If the bean is in the same xml config then use **<idref local="theTargetBean"/>**
- Use **<ref bean="theBeanId"/>** to refer to inject other beans into properties. To force order of creation you can use the depends-on="beanId"
- If you need lifecycle then use the **init-method** and **destroy-method** attributes. Do not do this on singleton beans. <bean id="egBean" class="foo" **init-method="init" destroy-method="cleanup"**/>
- Template beans avoid repitition; the abstract and parent attributes.
- Use ApplicationContext as your bean factory (e.g. FileSystemXmlApplicationContext)
- Do not use spring ApplicationListeners for broadcast of your own events – write your own mechanism and avoid being tied to spring. Event frameworks are very simple!
- **ApplicationContextAware** interface has method setApplicationContext() which the app context calls, allows the use of app contexts (resources, events etc) – but it ties you to spring!
- Data source beans are great, select your db pool implementation, define a datasource bean or set of them and inject it in.
- Leave the auto wiring mode alone. i.e. the default works out dependencies and creation ordering.

Client Code:

```
FileSystemXmlApplicationContext factory = new FileSystemXmlApplicationContext("config.xml");
```

BeanFactory methods:

```
boolean containsBean(String)
Object getBean(String) throws BeansException;
Object getBean(String,Class) throws BeansException;
boolean isSingleton(String) throws NoSuchBeanDefinitionException
Class getType(String name) throws NoSuchBeanDefinitionException;
String[] getAliases(String) throws NoSuchBeanDefinitionException;
```

Null param value null = <property name="email"><null/></property>
Empty string value "" = <property name="email"></property>

The list, set, map, and props elements allow properties and arguments of Java type List, Set, Map, and Properties, respectively, to be defined and set.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN" "http://www.springframework.org/dtd/spring-beans.dtd">

<beans>
      <!-- example of list and map and ref and property values -->
      <bean id="cityFinder" class="com.tallsoft.springeg.CityScape" singleton="true">
            <property name="cityMap">
               <map>
                  <entry key="LDN"><value>London</value></entry>
                  <entry key="FFT"><value>Frankfurt</value></entry>
               </map>
            </property>
      </bean>
      <!-- example of list and depends on for creation ordering-->
      <bean id="region" class="com.tallsoft.springeg.RegionInfo" depends-on="cityFinder">
            <property name="cityFinder"><ref bean="cityFinder"/></property>
            <property name="regions">
               <list>
                  <value>Europe</value>
                  <value>America</value>
               </list>
            </property>
      </bean>

      <!-- example of template, and the init method -->
      <bean id="templateGeography" abstract="true"
         class="com.tallsoftware.springeg.BaseGeography">
          <property name="planetName"><value>Earth</value></property>
          <property name="system"><value>sol</value></property>
      </bean>
      <bean id="earth" class="com.tallsoft.springeg.PlanetInfo" parent="templateGeography" />
      <bean id="mars" class="com.tallsoft.springeg.PlanetInfo"
                     parent="templateGeography" init-method="initialize">
            <!-- overriding the value here -->
            <property name="planetName"><value>Mars</value></property>
      </bean>
</beans>
```