**Spring Container**

* In a spring-based application, your application objects live in the Spring *container.*
* The container creates the objects, wires them together, configures them, and manages their complete lifecycle from cradle to grave (or new to finalize() ).
* The container is the core of the Spring Framework. Spring’s container used DI (dependencies Injection) to manage the components that make up an application. This includes creating associations between collaborating components. As such, these objects are cleaner and easier to understand, they support reuse, and they’re easy to unit test.
* There’s no single Spring container. Spring comes with several container implementations that can be categorized into two distinct types. *Bean factories* (defined by the *org.springframework.beans.factory.BeanFactory* Interface) are the simplest of containers, providing basic support for DI. *Application contexts* (defined by the *org.springframework.context.ApplicationContext* interface ) build on the notion of a bean factory by providing application-framework services, such as the ability to publish application events to interested event listeners.
* Although it’s possible to work with spring using either bean factories or application contexts, bean factories are often too low-level for most applications. Therefore, application context are preffered over bean factories.
* **Working with an application context:** 
  + Spring comes with several flavors of application context. Here are a few that you’ll most likely encounter:
    - *AnnotaionConfigApplicationContext-* Loads a Spring application context from one or more Java-based Configuration classes.
    - *AnnotationConfigWebApplicationContext –* Loads a Spring web application context from one or more Java-based configuration classes.
    - *ClassPathXmlApplicationContext –* Loads a context definition from one or more XML files located in the classpath, treating context-definition files as class path resources.
    - *FileSystemXmlApplicationContext –* Loads a contex definition from one or more XML files in the filesystem.
    - *XmlWebApplicationContext –* Loads context definitions from one or more XML files contained in a web application.
    - The difference between using *FileSystemXmlApplicationContext* and *ClassPathXmlApplicationContext* is that *FileSystemXmlApplicationContext* looks for knight.xml in a specific location within the filesystem, whereas *ClassPathApplicationContext* looks for knight.xml anywhere in the classpath (including JAR files).
    - Alternatively, if you’d rather load your application context from a Java configuration, you can use *AnnotationConfiApplicationContext.*
    - Instead of specifying an XML file from which to load the Spring application context, *AnnotatinConfigApplicationContext has been given a configuration class from which to load beans.*