Spring Boot offers a variety of features that simplify the development, deployment, and management of applications. Spring Boot’s approach spins around the “convention over configuration” idea, which minimizes the need for boilerplate code

Spring Boot is based on four core principles:

* Autoconfiguration -  it automatically configures the application based on the dependencies that are provided to it.
* Opinionated - it provides opinions about the best way to configure and run the application
* convention over configuration - provides acceptable default settings for configuration, which means that developers only need to provide configuration when they want to override the defaults. This simplifies the development process because developers do not need to spend time configuring the application themselves.
* eliminates boilerplate code -reduces the amount of boilerplate code that developers need to write by providing a set of defaults and abstractions.  For example, unlike JDBC, we don’t need to write a lengthy code just to get a connection with the database. Instead, we just need to provide values of database properties in 3-4 lines via the properties file to get a database connection.

**Spring Boot Application with Multiple Data sources**

In **application.properties** file set the configuration for two databases.

Note that the Song configuration prefixed with **(spring.datasource.\*)** and Lyrics configuration prefixed with **(spring.lyrics.datasource.\*)**, making the Song prefixed with (spring.song.datasource) would lead to error due to (spring.datasource.\*) not provided so at least one of the databases should act like if it was the default database.



create two config files for each datasource you have like **SongDBConfig.java** and **LyricsDBConfig.java,** **SongDBConfig.java**

since Song db is our **default** data source it is a must to include “**Primary**” annotation which defines that this bean going to be registered more than one time but I want this version to be the primary one. So, in the other class LyricsDBConfig.java the bean **won’t** have the “Primary” annotation.



**Actuator** to let you monitor and interact with your application, as well check the health status of the application

**Devtools** to have an embedded LiveReload server that can be used to trigger a browser refresh when a resource is changed

**Starter Web** to expose Rest Endpoints for the application

**Swagger** to visualize the API contracts from a web page and invoke the APIs as well

**Lombok** to replace boiler plate code like Getters Setters in classes with Nifty annotations

@ConfigurationProperties- annotation we used it to let spring boot know which configuration from application.properties should it use.

## options to deploy a Spring Boot Application?

1) Cloud Foundary  
2) Kubernetes  
3) Heroku  
4) OpenShift  
5) AWS (Amazon Web Services)  
6) Boxfuse and AWS  
7) Google Cloud