**Introduction to Spring Boot Framework**

Use Cases Manual

Sandbox Link [Spring Boot](https://share.percipio.com/cd/J676mja_Z)

**What is Spring Boot Auto Configuration?**

This guide will help you understand Spring Boot Auto Configuration with examples

**You will learn**

* Why do we need Auto Configuration?
* What is Auto Configuration?
* A few examples of Spring Boot Auto Configuration
* How is Auto Configuration implemented in Spring Boot?
* How to debug Auto Configuration?

**Tools you will need**

* Maven 3.0+ is your build tool
* Your favorite IDE. We use Eclipse.
* JDK 1.8+

**Why do we need Spring Boot Auto Configuration?**

**Spring based applications have a lot of configuration.**

When we use Spring MVC, we need to configure component scan, Dispatcher Servlet, a view resolver, web jars(for delivering static content) among other things.

<bean

class="org.springframework.web.servlet.view.InternalResourceViewResolver">

<property name="prefix">

<value>/WEB-INF/views/</value>

</property>

<property name="suffix">

<value>.jsp</value>

</property>

</bean>

<mvc:resources mapping="/webjars/\*\*" location="/webjars/"/>

**Below code snippet shows typical configuration of a Dispatcher Servlet in a web application.**

<servlet>

<servlet-name>dispatcher</servlet-name>

<servlet-class>

org.springframework.web.servlet.DispatcherServlet

</servlet-class>

<init-param>

<param-name>contextConfigLocation</param-name>

<param-value>/WEB-INF/todo-servlet.xml</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>dispatcher</servlet-name>

<url-pattern>/</url-pattern>

</servlet-mapping>

**When we use Hibernate/JPA, we would need to configure a data source, an entity manager factory, a transaction manager among a host of other things.**

<bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource"

destroy-method="close">

<property name="driverClass" value="${db.driver}" />

<property name="jdbcUrl" value="${db.url}" />

<property name="user" value="${db.username}" />

<property name="password" value="${db.password}" />

</bean>

<jdbc:initialize-database data-source="dataSource">

<jdbc:script location="classpath:config/schema.sql" />

<jdbc:script location="classpath:config/data.sql" />

</jdbc:initialize-database>

<bean

class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean"

id="entityManagerFactory">

<property name="persistenceUnitName" value="hsql\_pu" />

<property name="dataSource" ref="dataSource" />

</bean>

<bean id="transactionManager" class="org.springframework.orm.jpa.JpaTransactionManager">

<property name="entityManagerFactory" ref="entityManagerFactory" />

<property name="dataSource" ref="dataSource" />

</bean>

<tx:annotation-driven transaction-manager="transactionManager"/>

Spring Boot : Can we think different?

When a spring mvc jar is added into an application, can we auto configure some beans automatically?

**How about auto configuring a Data Source if Hibernate jar is on the class-path?**

**How about auto configuring a Dispatcher Servlet if Spring MVC jar is on the class-path?**

There would be provisions to override the default auto configuration.

Spring Boot looks at

a) Frameworks available on the CLASSPATH. b) Existing configuration for the application. Based on these, Spring Boot provides basic configuration needed to configure the application with these frameworks.

This is called **Auto Configuration.**

Bootstrapping REST Services with Spring Initializr

Creating a REST service with Spring Initializr

Spring Initializr <http://start.spring.io/> is great tool to bootstrap your Spring Boot projects.

Launch Spring Initializr and choose the following

Choose com.optum.springboot as Group

Choose student-services as Artifact

Choose following dependencies

Web

Actuator

DevTools

Click Generate Project.

Import the project into Eclipse. **File -> Import -> Existing Maven Project**.

Add a course to an existing student