**Before moving further please go through following things very carefully :**

1. Theoretical part of Spring Data JPA from our notes

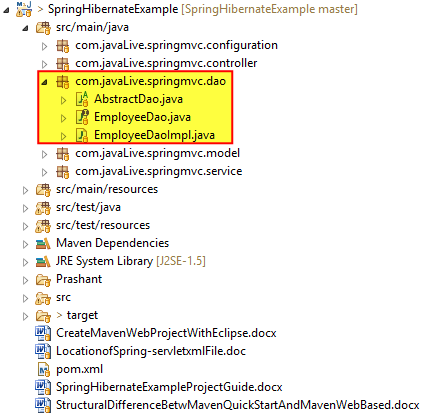
2. Our Spring MVC application viz. 'SpringHibernateExample' with its package structuring and each file in detail for better understanding of this project as this project is nothing but replacement of its one component that's it.

3. You must know creation, structure and functioning of normal Spring Boot application. Please refer project 'SampleSpringBootProject' along with its project guide and other .docx file in the project.

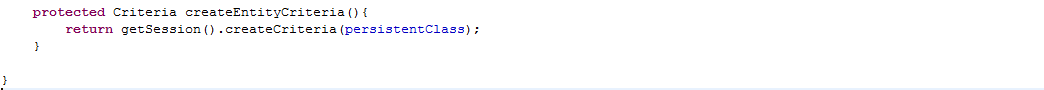
*Please note that in this project guide we are not going to explain each and every thing in detail, we are just comparing our previous project of Spring MVC viz. 'SpringHibernateExample' with Java configuration and using this comparative analysis you have to understand what is Spring Data JPA*

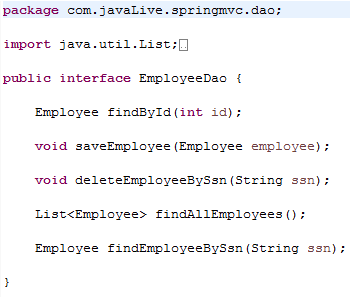
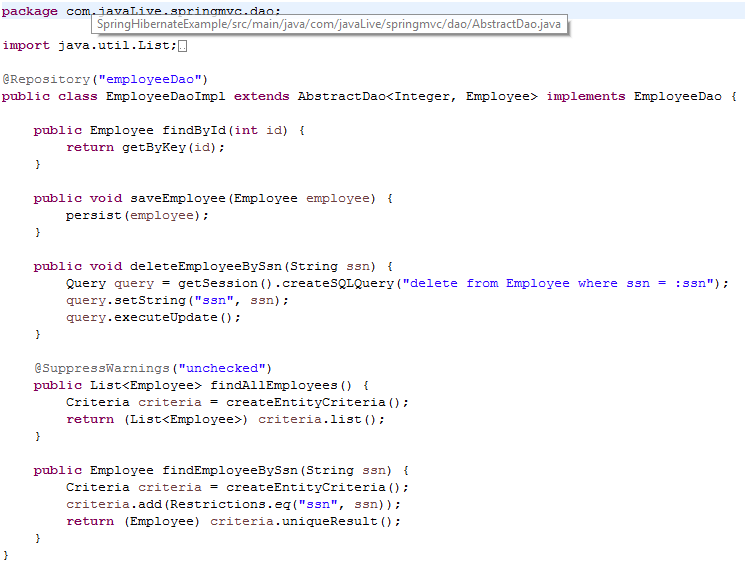
**\*\*\*\*\*SpringDataJPA is nothing but to replace DAO layer code with single interface extending JPARepository interface.**

**Now below is the project which is plain old way without Spring Data JPA:**

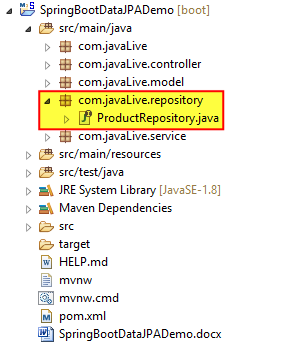


We all knows that DAO is the layer which is responsible for interacting with database but during this process we require lots of code among which most of it is redundant. Please refer below screen shots for more details.

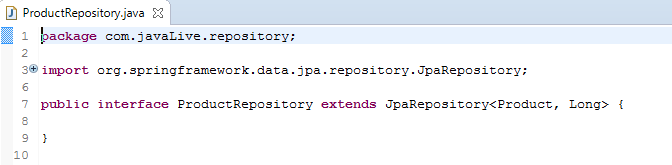
 

Now if we add any other entity class then again we have to repeat these files for it also. To provide solution for this problem. Spring provides a concept called Spring Data JPA which provides a interface viz. **JPARepository**, we have to simply extend that interface and that's it, Spring Data JPA provide implementation for all the methods those are generally used in DAO layer. These methods are predefined in JPARepository interface and its implementation is provided by Spring Data JPA behind the scene. Please refer below screen shot:



The details of the highlighted fillies as below:

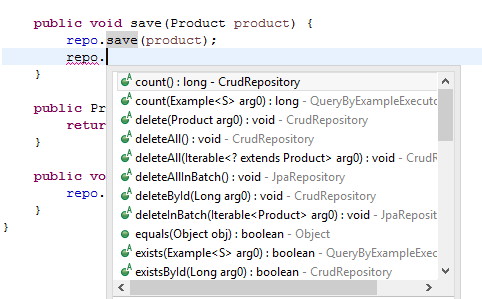
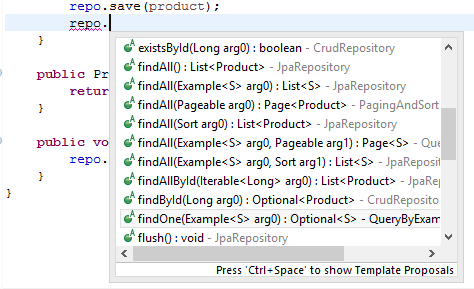
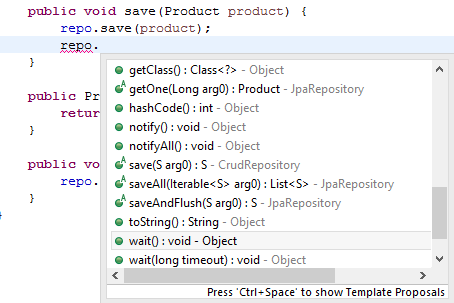


Please note that we haven't written a single line of code here.



Now carefully go through above screen shot where we are using predefined and pre implemented code of Spring Data JPA. Note the line of code before and after using Spring Data JPA and that's the magic.

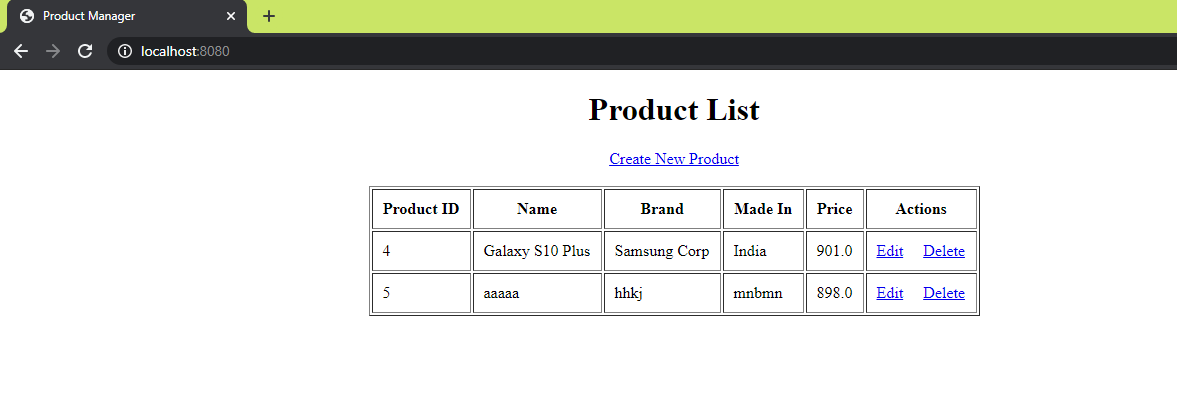
Below screen shot shows methods with pre implementation of JPARepository interface.

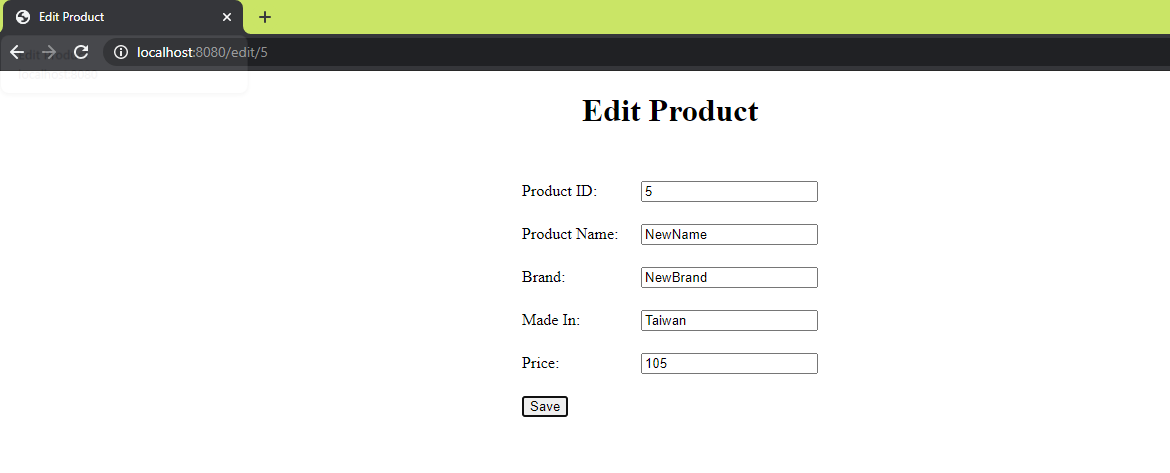
Please note that this is applicable for both regular Spring MVC application and also for Spring Boot.

**Running the project:**

Right click on com.javaLive.SpringBootDataJpaDemoApplication->Run As-> Spring Boot App



Click on 'Edit' button of second row:



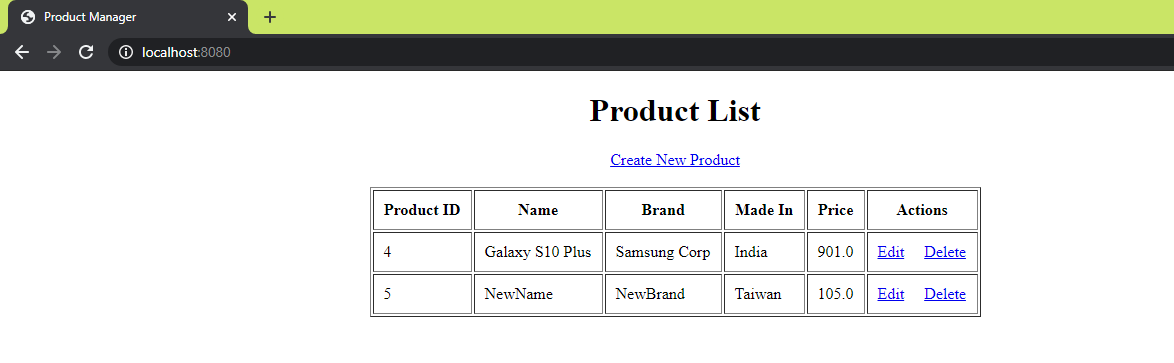
Click on 'Save'



Click on 'Create New Product'



Now click on Delete link of last product:



**What Is the Difference Between Hibernate and Spring Data JPA?**

Hibernate is a JPA implementation, while Spring Data JPA is a JPA Data Access Abstraction. Spring Data offers a solution to GenericDao custom implementations. It can also generate JPA queries on your behalf through method name conventions.

With Spring Data, you may use Hibernate, Eclipse Link, or any other JPA provider. A very interesting benefit is that [you can control transaction boundaries declaratively using the @Transactional annotation](https://vladmihalcea.com/a-beginners-guide-to-transaction-isolation-levels-in-enterprise-java/).

Spring Data JPA is not an implementation or JPA provider, it's just an abstraction used to significantly reduce the amount of boilerplate code required to implement data access layers for various persistence stores. I hope this article useful to you. You can find more answers to this question at [StackOverflow](https://stackoverflow.com/questions/23862994/what-is-the-difference-between-hibernate-and-spring-data-jpa/23863416" \l "23863416" \t "_blank).

Hibernate provides a reference implementation of the Java Persistence API that makes it a great choice as an ORM tool with benefits of loose coupling.

Remember, Spring Data JPA always requires the JPA provider such as [Hibernate](http://www.javaguides.net/p/hibernate-tutorial.html) or Eclipse Link.