DOWNLOAD PDF SPRING 4 AND HIBERNATE 4

Chapter 1 : Maven + Spring + Hibernate + MySql Example â€“ calendrierdelascience.com

Spring 4 provides support for Hibernate 4's SessionFactory through a LocalSessionFactoryBean which is actually a FactoryBean that creates a Hibernate's SessionFactory which is then injected to Hibernate-based DAO beans. Here's the bean declaration.

Spring framework provides an implementation - the SpringServletContainerInitializer class which delegates a Servlet context to any implementations of the WebApplicationInitializer interface. The actual class that does the configurations is ApplicationContextConfig, which is covered in section 4 below. Now we extend it for full CRUD operations. Hence, update the UserDAO interface as the following code: Configuring Spring Application Context using Java-based Configuration Now, we come to the most important and interesting part of the application, which configures beans definitions using Java code instead of XML. Create ApplicationContextConfig class with the following Spring annotations: The ComponentScan annotation tells Spring to scan the specified package for annotated classes the HomeController class in case of this tutorial. Remember to change attributes of the DataSource according to your environment. Spring Integration in Action help you learn more about enterprise integration and messaging using the Spring Integration framework. The following method configures a SessionFactory bean: Notice this statement tells Hibernate to load the User class into its mapping definitions: Instead, the Transactional annotation is used to tell Spring automatically inserts transaction management code into the bytecode. The following method configures a bean which is a UserDAO implementation: Discover how to write efficient batch applications with Spring Batch in Action 7. Click New User link to add a new user. Enter some dummy information then click Save. The user is saved and we go back to the users list page: Now we can click Edit or Delete link to update or remove a user. You have completed our second part of the Spring-Hibernate integration series. For your convenience, we provide downloads for the project and a deployable WAR file in the attachments section.

Chapter 2 : Spring 4 and Hibernate 4: Agile Java Design and Development by Amritendu De

To add the Spring Persistence dependencies to the project calendrierdelascience.com, please see the article focused on the Spring and Maven dependencies. To use Hibernate 4 in a project, a few things have changed on the configuration side when moving from a Hibernate 3 setup. The main aspect that is different when.

Reading List Java 8 shipped with new language features and libraries and Spring 4. This article will walk you through the new Java 8 features that are supported by Spring 4. Spring 4 supports Java 6, 7, and 8 Code compiled with the Java 8 compiler produces. Since Spring makes heavy use of reflection and other byte code manipulation libraries such as ASM, CGLIB and others, it was important to make sure that those libraries were able to understand the new Java 8 class files. Related Vendor Content Related Sponsor The Spring framework itself is compiled using the Java 8 compiler with the command line option to produce Java 6 byte code. Therefore you can compile and run Spring 4. Spring and Java 8 Lambda expressions The designers of Java 8 wanted to ensure that it was backward compatible so that Java 8 lambda expressions can be used with code compiled with previous versions. Backward compatibility was achieved by defining the concept of functional interfaces. Basically, the Java 8 designers analyzed existing bodies of Java code and noticed that most Java programmers used interfaces with a single method to represent the idea of a function. With Java 8 we can write a lambda expression as the value for the second argument of the query method. Instead of writing code like this: Covering all the details of Java 8 functional interfaces is beyond the scope of this article and it is highly recommend you learn the details of functional interfaces elsewhere. However the key take away point is that Java 8 lambda expressions can be passed to methods compiled with Java 7 and earlier if the methods accept a functional interface. The Spring code base contains a lot of functional interfaces, therefore lambda expressions can be easily used with Spring. Even though the Spring framework itself is compiled to Java 6. In conclusion, because the Spring Framework has been using functional interfaces before Java 8 formalized the definition of a functional interface it is easy to use lambda expressions with Spring. The new Date and Time API is worthy of its own article therefore we will not cover it in detail other than to note that the new java. Spring ships with a data conversion framework that is able to convert from strings to Java data types and from Java data types to strings. Therefore you can write code like this. While Spring 4 supports the Java 8 date and time libraries, that does not mean that third party frameworks like Hibernate and Jackson are able to support the Java 8 date and time. Spring 4 and Repeating Annotations Java 8 added support for repeating annotations and Spring 4 supports these. In particular Spring 4 supports repeating Scheduled and PropertySource annotations. For example, notice the repeating PropertySource annotation in the code snippet below. One way to eliminate NullPointerExceptions is to make sure that methods always return a non null value. For example consider a method like the following: Optional class makes it possible to write our interface like this. The method signature and compiler will help make it clear that a value is Optional. A more detailed description of the idea of Optional class be found here. In summary the Java 8 Optional class makes it easier to write code with fewer NullPointerException bugs and Spring works nicely with the Java 8 Optional class. The Java 8 complier will write out argument names into the. Before Java 8, Spring was able to extract parameter names from the compiled code if the code was compiled with â€”debug option. Therefore for projects like Spring Data that auto generate repository implementations based on Java interfaces this caused a problem. For example consider the interface below. With Java 8 â€”parameters option Spring Data is able to discover the names of parameters on abstract methods so we can rewrite the interface as. So when using Java 8 it is a good idea to compile your code with the â€”parameters flag. Conclusion Spring 4 runs on Java 6, 7, and 8 and as a developer you can write your application code using any of Java 6, 7, 8. If you are using Java 8 you will be able to make use of lambda expressions to write cleaner more compact code anywhere there is a functional interface. Since Spring uses a lot of functional interfaces there are a lot opportunities for using lambda expressions with Spring 4. Java 8 shipped with improved libraries such as the new java. Finally, compiling Java 8 code with â€”parameters option preserves argument names in methods and makes it possible to write more compact Spring MVC method handlers and Spring Data query methods. If you are ready to start using Java 8 in your projects you will find that Spring 4 is a good framework that makes use of the features of Java 8. About the Author Adib Saikali is a Senior Field Engineer at Pivotal with a passion for technology and entrepreneurship from assembly to JavaScript from cold calling to pitching venture capitalists. You can connect with Adib on twitter asaikali.

Chapter 3 : Hibernate4 + spring4 + maven example - My Cute Blog

This article is an example of java application showing the integration of Spring 4 and hibernate 5. It explains how to configure Hibernate 5 and Spring 4 along with transaction manager to perform database operations.

How Spring supports Hibernate Integration Basically, in order to support Hibernate integration, Spring provides two key beans available in the org. Programmers can use Transactional annotation in DAO methods to avoid writing boiler-plate transaction code explicitly. Setting up Database Execute the following MySQL script in order to create a database named usersdb with a table named users: Project Structure The following screenshot shows final structure of the project: After completing this tutorial, you will create a project structure looks like the above. Maven Dependencies Declare versions for Java and Spring framework: You can see the whole content of pom. For more information about Hibernate XML mapping, see: The list method simply obtains the current session from the SessionFactory and queries for a list of all users in the database. Change database URL, username and password according to values in your environment. This data source will be injected to a SessionFactory bean below. The configLocation property specifies where Hibernate configuration file will be searched for. In this case, it is the hibernate. Configuring TransactionManager Bean The following declaration is for automatic transaction support for the SessionFactory: For the whole content of Spring application context configuration file, see the corresponding file in the attached project. It is injected to this controller automatically so that the handling method home can use it to list all users from the database. And eventually, the home method returns a view named home which is resolved an actual JSP page which is described below. Coding View Page Create a home. You have completed our first part of Spring-Hibernate Integration series. You can download the attached project and experiment yourself. A deployable WAR file is also provided for your convenience.

Chapter 4 : Spring 4 and Hibernate 4 Integration Tutorial Part 1: XML Configuration

In this tutorial, we will integrate Spring 4 with Hibernate 4 using annotation based configuration. We will develop a simple CRUD java application, creating hibernate entities, saving data in MySQL database, performing database CRUD operations within transaction, and learn how different layers interacts with each-other in typical enterprise application, all using annotation based configuration.

Gradle and Maven to Build the Project Find the gradle file to build the project. We will create a table for user profile that will contain columns such as user name, password, role etc. Find the database schema. It can be maximum 72 characters long, so we should keep our password column accordingly. I am using password column length as If we do not set the proper length of password column then while saving BCrypt encoded password, it will be truncated and hence the user will never be authenticated because of wrong password. To create BCrypt password, we can use a simple main class. We will create separate JavaConfig files for spring MVC, spring security and spring database configuration using hibernate. We have service layer, DAO layer and controller. For login, we will create a custom login page. Let us start step by step. Find the configuration file. These files can only be accessed by controllers using InternalResourceViewResolver. It has methods such as setPrefix and setSuffix. EnableWebMvc is annotated at class level with Configuration annotation. ComponentScan annotation is used to configure component scanning directive. Here we have specified the directive as com. All the components within this package and its child packages will be scanned. Now we need to register DispatcherServlet. When we are using JavaConfig then preferred way to register DispatcherServlet is as follows. To override any security method we need to extend WebSecurityConfigurerAdapter class. We have overridden configure method from WebSecurityConfigurerAdapter class. Within this method we are performing spring security login, logout and exception handling configuration. If we want to disable it we can do it using http. To authenticate user, we will use custom UserDetailsService which in turn will use hibernate ORM to interact with database. Using configureGlobal we will configure our custom UserDetailsService instance. For this purpose spring security provides AuthenticationManagerBuilder. Here we also configure password encoder. In our example we are using BCryptPasswordEncoder for password encoding. It uses BCrypt strong hashing function. While instantiating BCryptPasswordEncoder we can optionally pass strength or we can also pass SecureRandom instance to its constructor. The default strength is In our example we are using Secured annotation to secure service layer. We can do it as follows. Find the spring security initializer. For hibernate transaction management, spring has HibernateTransactionManager class. It allows one thread-bound session per factory. To enable HibernateTransactionManager we have to create a bean of it in our configuration file. To create a data source, we are using third party library commons-dbcp2. For basic implementation we are using BasicDataSource class of commons-dbcp2. We need to instantiate this class and set the database configuration related value such as driver class name, URL, username and password. In our example we are using MySQL database. To configure hibernate properties we have created a method that returns java Properties loaded with hibernate properties such as hibernate. Using LocalSessionFactoryBean we configure data source, packages to scan entity and hibernate properties. Create a bean of HibernateTemplate. This is a helper class that is used to interact with database. It automatically converts hibernate exceptions to data access exceptions. PropertySource annotation accesses the property file and data is loaded to Environment and then by calling its method getProperty we fetch properties defined in property file. Now find the property file used in our example. Database username and password values are our local values for demo. You need to change it according to your database envoromnet. We need to create a class and implement UserDetailsService for user authentication. It has a method loadUserByUsername that accepts user name and returns org. Here we are using hibernateTemplate. Find the DAO class. Transactional annotation is used make DAO methods transactional. This annotation can be used at class level as well as method level. It has attributes to define propagation, transaction isolation level etc. We can configure these values as follows. Now find our service layer. To make a method secure, spring has Secured annotation. Secured annotation is effective only if we use below annotation in our spring security configuration class. The method home will be called after successful authentication and the method error will be called when an authenticated user tries to access service layer secured method and throws access denied exception. Create Views Find the spring security login page. We have created a custom login page with CSRF protection. After successful login, a user will access service layer. If the user role is authorized to access service layer, then user will get page which will display user information. Find the JSP page that will be used to display user information. Service layer, DAO layer, controller and views will be same as given in the application using annotation. Let us start now to understand XML configuration. We need to specify base package. All the components of this package and sub packages will be scanned. Annotation configuration will be true by default. Specify prefix and suffix that will be used to create URL for views. Create a session factory with LocalSessionFactoryBean using data source and hibernate properties. Using session factory create a bean for transaction manager with HibernateTransactionManager. Using session factory we will also create a bean for HibernateTemplate that will be used to query the database. Now configure transaction manager for annotation driven transaction management. Now configure ContextLoaderListener to listen startup and shutdown spring root web application context. We will also configure DelegatingFilterProxy filter that works as a proxy for standard servlet filter delegating to spring-managed bean. Now all calls to filter will be delegated to that bean in spring context. Run Application To build and run the demo application, follow the steps. Import database in MySQL given above on this page. Download the source code from the link given below on this page. Go to the root directory of the project using command prompt. Build the project using gradle with following command. If we want to build the project using maven, use following command. Find the print screen. User will be able to see profile. Now login using this credential. Find the print screen of the output. I am done now. Happy Spring Security learning!

Chapter 5 : Spring Hibernate Integration Example (Spring 4 & Hibernate 3) â€“ TechnicalStack

In this second part of the Spring and Hibernate integration tutorial series, we demonstrate how to develop a Spring MVC - Hibernate application without using any XML configuration.

We are using spring boot to resolve spring JAR dependencies. DataSourc and is used for basic requirements. Annotation to use configuration file. It represents the application environment using which we can fetch property values. Helper class to execute hibernate methods. It binds a hibernate session from the specified factory to a thread. It creates hibernate SessionFactory. It enables annotation driven transaction management in spring. Now find the database property file. It is MessageSource implementation which accesses resources using specified basename. It serves static resources. It helps in stateless application to resolve locate. It sends back a cookie to the user in the form of time zone etc to resolve locale. It has prefix and suffix property that is used to access the file. Web Application Initializer Find the web application initializer. It is supported by Servlet 3 and we need to use web. Using this annotation, class becomes controller. The property annotated with this annotation is validated. It is used for model data for UI as java Map. It provides validation error object. It create HTML form. It creates HTML input field. It creates HTML password field. It creates radio button. It creates combo box. It prints HTML field validation error. It enables annotation based transaction management. It scans the annotated component in the given classpath specified in base-package. It enables annotation based spring MVC controller. It is used to serve static resources. We need to define mapping for actual resource path. Create database using MySQL. Navigate to root directory using command prompt and run maven command as mvn clean package. Install war file in Tomcat 8 5. Run the URL http: To add the person, fill the data and then click on Add button. To update a person, first click on Edit and do changes and then click on Update button. To delete a person, click on Delete button. Now I am done. Happy spring MVC learning! Download Complete Source Code.

Chapter 6 : Spring 4 + Hibernate 5 Example - BytesTree

In this tutorial I will show you how to integrate Struts 2, Spring 4, Hibernate 4 and Maven. In the previous example Integrate Spring 3, Struts 2 and Hibernate 3, I have shown how to integrate Struts 2, Spring 3 and Hibernate 3 but I have not used maven there.

You have done a great job writing so complex ideas on a so simple way with great examples. I am a fan of your website. I hope to see more Spring Hibernate with Spring Security combo examples which mimics real world application. Amila Viraj Good tutorialâ€¦ I just want to develop a spring mvc project. And also after entering user details, i want to generate a pdf using that details. You have done a phenomenal job writing so complex ideas on a so simple way with great examples. Once you got the idea, One-to-Many would be even simpler to implement. I am always refer your side to learn java related technologies. Thank you very much Sirâ€¦â€¦â€¦â€¦â€¦â€¦..!!! Bhuman Soni Good stuff! PdfWriter cannot be resolved. It is indirectly referenced from required. You can find it at http: Thank you so much. In I had a doubt in spring MVC.. Can you please explain which scope we will configure to the controller in real world application. How a controller will handles the more requests I mean how many objects will be created for controller.. Please help me on thisâ€¦My mail mail ID is madhusitams gmail. Keep it up guys thank you for what you are doing! Ali Honarmand Thank you very much for the detailed examples. They helped me a lot. Nishanth Raj This is very good to share spring 4 mvc knowledge. Please put some more concepts of spring 4 mvc and examples for that. Thanks in advanceâ€¦ Marcos Vinicius Moraes Gabriel.

Chapter 7 : java - Spring 4 + Hibernate 4 configuration - Stack Overflow

Step 2 Create a configuration file (if you are using eclipse create this file in src folder and if this is web application it can go anywhere in WEB-INF but need to configure the same in calendrierdelascience.com) to tell Spring context factory what packages it need to look for beans configured using annotations.

Chapter 8 : Spring 4 & Hibernate 4 integration tutorial

I use Spring RELEASE + Hibernate calendrierdelascience.com configuring and starting my web-app on Tomcat 7 I have sessionFactory == null.. Can you write what I did wrong? Here is my sources.

Chapter 9 : Spring 3 and Hibernate 4 Integration Example Tutorial - HowToDoInJava

Spring 4 MVC + Hibernate 4 + MySQL + Maven CRUD Integration using Annotation and XML with Tomcat 8 and Spring Boot Example By Arvind Rai, May 20, This page will walk through Spring 4 MVC, Hibernate 4, MySQL, Maven CRUD integration using annotation and XML with Tomcat 8 and spring boot example.