DOWNLOAD PDF JAVA WEB APPLICATION DEVELOPMENT TUTORIAL

Chapter 1 : Web Tools Platform

Java Web Application is used to create dynamic websites. Java provides support for web application through Servlets and racedaydvl.com can create a website with static HTML pages but when we want information to be dynamic, we need web application.

Sure, building traditional desktop and even mobile applications is all well and fine. But what if you want to leave your current background behind and start stepping on the web terrain? The good news is that the language ships with the fully-fledged Servlet API, which lets you build robust web applications without much hassle. A Quick and Dirty Introduction to Java Servlets Enter servlets , a specific type of Java program executed within the scope of a web container also called servlet containers, Tomcat and Jetty are prime examples , which allows to handle client requests and server responses in a straightforward and performant fashion. Suffice it to say that servlets are instantiated and destroyed by their containers instead of the developer and act as a middle layer between clients usually web browsers and other applications running on the server databases, for example. The best way to get started using servlets is with a concrete example. Data submitted by customers will be collected by a servlet and validated at a very basic level through some static helpers. But first things first. The first step we need to take to build the application is to define the so-called deployment descriptor DD. As the name suggests, it specifies how an application should be deployed in a specific environment. Not surprisingly when it comes to web applications, the deployment descriptor is a plain XML file called web. It can of course hold a lot more content, including servlet mapping directives , intialization parameters , a list of welcome files and a few additional settings. However, in its current state the application literally does nothing. Since the primary goal here is to let customers sign up using an HTML form, the next thing we need to do is to define the JSP files tasked with rendering the aforementioned form and with displaying customer data once the signup process has been successfully completed. The first one will be responsible for rendering the signup form and for displaying eventual errors triggered after validating the inputted data. The second one will be a typical welcome page, which will show the data submitted by the customer after successfully completing the signup process. This is achieved thanks to the functionality of objects accessible from within a JSP file, such as request. We will see below how the servlet is mapped to the processcustomer URL and how it controls the inputted data. With the presentational layer already set, the next step is to create the servlet responsible for collecting customer data from POST requests and validate the data in a basic fashion. Building a Customer Controller Defining a servlet capable of grabbing the data entered in the previous signup form is a breeze, trust me. All that we need to do is subclass the native HttpServlet class and implement its doGet or doPost methods or both when required. In this particular case, the customer servlet will intercept data coming from POST requests. Its definition is as follows: Just plain class-level annotations handle the mapping process for us. Otherwise, some servlet containers will fail to do the mapping and throw an ugly HTTP status error, telling you that the requested resource the servlet itself is not available. That said, the CustomerController class itself performs a few simple tasks. First of all, it collects the data entered in the signup form by using an implementation of the native HttpServletRequest interface, which holds the values corresponding to the firstname,lastname and email fields of the signup form. Secondly, it sets these values as request attributes, so they can be redisplayed either in the signup form view, or in the result view. Lastly, the data is validated by using a couple of static helpers, which check for empty strings and well-formed email addresses. The validators themselves are simple classes, checking a couple of desired properties, for example whether the name is non-empty and the email address looks like, well, an email address. If you want to know how they are implemented, you can look at them here. Otherwise, the welcome page is rendered, as the customer has successfully signed up. Maybe the biggest pitfall here is the use of static helpers for checking customer data instead of appealing to a standard approach, such as validating domain objects and even entire object graphs by using Java Bean Validation. The details on how to do this will be left as the topic of an upcoming article. To do so, just follow the steps below: As prerequisites you need Git make sure to pick up the version that works with your operating system , Maven , and a servlet container among others, Apache

Tomcat , Jetty , or JBoss Wildfly. It is possible that your IDE ships with one or more of these systems, in which case you can use the embedded versions. Use Git to clone the application repository to your local machine and import it into your IDE, preferably as a Maven project. Deploy the whole project to the servlet container and run it. Feel free to pat yourself in the back. Final Thoughts At this point you have acquired all the skills required to kick off building your own Java web application and, best of all, without having to resort to the complexities of any framework at all. On one hand it shows in a nutshell how easy it is to handle request parameters and send responses back to the client in different formats. But this functionality comes at a price: This is not inherently bad, as the locators are just data holders. If you want to give the customer application a try and play around with it, just clone the repository from GitLab. Feel free to leave me your comments. He has more than 12 years of experience in PHP development, 10 years in Java Programming, Object-Oriented Design, and most of the client-side technologies available out there.

Chapter 2 : Java application development tutorial using Azure Cosmos DB | Microsoft Docs

A Java web application is a collection of dynamic resources (such as Servlets, JavaServer Pages, Java classes and jars) and static resources (HTML pages and pictures). A Java web application can be deployed as a WAR (Web ARchive) file.

You will need a WTP runtime. Once you have everything downloaded, install JDK and your favorite server runtime, and unzip eclipse, emf, gef, jem and wtp zip files to a folder. You are ready to go! Setup Preferences After you launch eclipse for the first time, but before we can build our first web application, we need to do a little eclipse house keeping: If it is not already defined, you will need to set the Java preferences to point to your JDK installation see figure. You should have obtained the latest JDK from Sun. During web application development, we will need a server runtime environment to test, debug and run our project. We begin by telling wtp what our server runtime is, and where it is located. A server is an instance of the server runtime that can host our web applications and other server-side components. To define a server runtime, we need to visit the appropriate Preferences page: Here you will find a list of server runtimes that have been defined previously. You can choose a server runtime and change its properties. A wizard will popup and display a list of server runtimes that are supported by WTP. Choose your server runtime from the list. It does not provide a server instance that can be used to run web artifacts. Of course, based on your choice of the server runtime, you will be asked to provide different properties. Now we can do the fun part. We chose Apache Tomcat. You can also choose another server such as ObjectWeb Jonas. There are differences between for server runtimes; Apache Tomcat provides dynamic development support; i. Generic server does not support dynamic development, but it will automatically publish the artifacts your web application to the server when you run. Choose Web category from the list of available wizards to create a new eclipse resource. You will find two project types available under this category. Simple Web Project is a basic Eclipse resource project that can be associated with a server. It needs to be associated with a Target Server. A target server provides the container which will be used to execute our web application. Specifically, it will provide a set of libraries jars , that are added to the project classpath, which are needed to compile our custom classes. These are not a part of the JDK libraries. Server runtime provides them to the project. If yiu have skipped the previos step where we defined a server runtime, you will get a chance to define one here by clicking on the Newâ€¦. We will create a standalone web application that can be deployed as a standard web module. This is used when you access the web application with a URL, i. It has created a java source folder. You will add your custom java packages and classes here. It will contain the web resources that will be packaged with your J2EE web module. All artifacts inside this folder are accessible from the Web Application context. Create a new Java class that is a subclass of the standard javax. Add the code provided in Listing 1 to the servlet: This deployment descriptors is called the web. Enter the following lines into web.

Chapter 3 : Building a Web App with Java Servlets â€” SitePoint

Development of presentation-oriented web applications is covered in Chapter 4, JavaServer Faces Technology through Chapter 9, Developing with JavaServer Faces Technology. Service-oriented: A service-oriented web application implements the endpoint of a web service.

To take advantage of this, you need to configure a JDBC Java Database Connectivity data source for the server which your application can use for connection pooling. You could configure the data source directly within the GlassFish server Admin Console, or, as described below, you can declare the resources that your application needs in a glassfish-resources. When the application is deployed, the server reads in the resource declarations, and creates the necessary resources. The following steps demonstrate how to declare a connection pool, and a data source that relies on the connection pool. Optionally, add a description for the data source. For example, type in: Click Next, then click Next again to skip step 3, Additional Properties. Make sure the Extract from Existing Connection option is selected, and choose jdbc: The wizard detects any database connections that have been set up in the IDE. Therefore, you need to have already created a connection to the MyNewDatabase database at this point. In Step 5, select javax. Note that the IDE extracts information from the database connection you specified in the previous step, and sets name-value properties for the new connection pool. The wizard generates a glassfish-resources. In the Projects window, you can open the glassfish-resources. The server starts up if not already running, and the project is compiled and deployed to it. Note that the new data source and connection pool are now displayed: Referencing the data source from the application You need to reference the JDBC resource you just configured from the web application. Deployment descriptors are XML-based text files that contain information describing how an application is to be deployed to a specific environment. For example, they are normally used to specify application context parameters and behavioral patterns, security settings, as well as mappings for servlets, filters and listeners. In the Projects window, expand the Configuration Files folder and double-click web. Click the References tab located along the top of the editor. The Description field is optional, but you can enter a human-readable description of the resource, e. The new resource is now listed under the Resource References heading. To verify that the resource is now added to the web. Select the GlassFish server in the left pane. Before you close the Servers manager, make a note of the path indicated in the Domains folder text field. When you connect to the GlassFish server in the IDE, you are actually connecting to an instance of the application server. Each instance runs applications in a unique domain, and the Domain Name field indicates the name of the domain your server is using. As shown in the image above, the driver JAR file should be located within domain1, which is the default domain created upon installing the GlassFish server. Click Close to exit the Servers manager. If you do not see the driver JAR file, perform the following step. Deploy your project to the server. Adding Dynamic Logic Returning to the index. To do so, perform the following three tasks. You can verify this by expanding the GlassFish Server node under the Libraries node in the Projects window, and searching for the javax. Older versions of the GlassFish server use the jstl-impl. JSTL provides the following four basic areas of functionality. Implementing JSTL code Now you can implement the code that dynamically retrieves and displays data for each page. Both pages require that you implement an SQL query that utilizes the data source created earlier in the tutorial. Hover your mouse over the DB Report item in the Palette. In the dialog that displays, enter the following details: The following content is generated in the index. New content shown in bold. A taglib directive declares that the JSP page uses custom i. Run the project to see how it displays in a browser. Right-click the project node in the Projects window and choose Run. When you choose Run, the IDE deploys the project to the GlassFish server, the index page is compiled into a servlet, and the welcome page opens in your default browser. The code generated from the DB Report item creates the following table in the welcome page. As you can see, the DB Report item enables you to quickly test your database connection, and enables you to view table data from the database in your browser. This can be particularly useful when prototyping. The following steps demonstrate how to integrate the generated code into the HTML drop-down list you created earlier in the tutorial. Examine the column data in the generated code. This causes the JSP container i. In this manner, data for the entire table is displayed. Changes are displayed in bold. Delete the table that was generated from the DB Report item. Deletion shown below as strike-through text. Dec 22, , 7: Refresh the welcome page of the project in your browser. Note that the drop-down list in the browser now contains subject names that were retrieved from the database. You do not need to redeploy your project because compile-on-save is enabled for your project by default. This means that when you modify and save a file, the file is automatically compiled and deployed and you do not need to recompile the entire project. You can enable and disable compile-on-save for your project in the Compiling category of the Properties window of the project. Enter the following details in the Insert DB Query dialog box. The following content is generated in the response. Dec 22, , 8: Recall that in index. Changes below shown in bold: Try running it again to see how it displays in a browser. When you run a project, you can be sure the deployment contains your latest changes. Click the Run Project button in the main toolbar. You should now be forwarded to the response. This document demonstrated how to create a simple web application that connects to a MySQL database. If your application does not display correctly, or if you are receiving a server error, the following examinations may be useful.

Chapter 4 : Web Application Development in Java | Tutorial

Eclipse Web Tool Platform (WTP). This tutorial describes the development of servlets and Java web application with Eclipse WTP. The Eclipse WTP project provides tools for developing standard Java web applications. Typical web artifacts in a Java environment are HTML, CSS and XML files, webservices.

An HTML form is added to the index. Delete the value attribute from this tag. Specify the following values: Type Enter your name: Name the file response, and click Finish. Notice that a response. The Insert Use Bean dialog opens. Specify the values shown in the following figure. Otherwise, it overwrites the value for name that you pass in index. In this case, the user input coming from index. Therefore, by setting property to name, you can retrieve the value specified by user input. Specify the following values in the Insert Get Bean Property dialog: Property names are case-sensitive. The "name" property must be in the same case in response. When you run a web application, the IDE performs the following steps: Building and compiling the application code see note below. You can perform this step in isolation by selecting Build or Clean and Build from the project node context menu. Deploying the application to the server. You can perform this step in isolation by selecting Deploy from the project node context menu. Displaying the application in a browser window. By default, the project has been created with the Compile on Save feature enabled, so you do not need to compile your code first in order to run the application in the IDE. The IDE opens an output window that shows the progress of running the application. Look at the HelloWeb tab in the Output window. In this tab, you can follow all the steps that the IDE performs. If there is a problem, the IDE displays error information in this window. The IDE opens an output window showing the server status. Look at the tab in the Output window with the name of your server. If the GlassFish server fails to start, start it manually and run the project again. You can start the server manually from the Services window, by right-clicking the server node and selecting Start. The server output window is very informative about problems running Web applications. Note that the browser window may open before the IDE displays the server output. Enter your name in the text box, then click OK. When I click the OK button for index. What error messages are there? What JDK does your project use? JDK 7 requires GlassFish 3. The server version is in the Run category. Lastly, download the sample project and compare it with your own. This overwrites the value you passed in the index. Delete the value attribute. Is the server running? Was the application deployed? If the server is running and the application was deployed, are you getting an org. This usually means that a value in your code is not initialized correctly. In this tutorial, it means that you probably have a typo somewhere in a property name in your JSP files. Remember that property names are case-sensitive! This document demonstrated how to create a simple web application using NetBeans IDE, deploy it to a server, and view its presentation in a browser. It also showed how to use JavaServer Pages and JavaBeans in your application to collect, persist, and output user data. For related and more advanced information about developing web applications in NetBeans IDE, see the following resources: Introduction to the Struts Web Framework.

Chapter 5 : Introduction to Java Web development - Tutorial

In web application mainly GET and POST method is used. GET method is used to ask for data from server while POST method is used to submit data to server. GET method can also send data to server but in that case data will be appended to the end of URL.

Create a new Dynamic Web Project called com. Press twice the Next button and select the Generate web. Afterwards press the Finish button. A new project has been created with the standard structure of a Java web application. Creating Data Access Object Create a new package called com. Create the following new Java class to read and write the counter value to and from the file. Enter the following data. You could also create a servlet without the wizard. The wizard creates a Java class which extends the javax. HttpServlet and adds the servlet settings to the web. Enter the following code. The servlet will increase the counter if the user was inactive for 5 seconds. You application must be deployed to the server and started there. Ensure your servlet is selected to run on the server. Check the server for errors. Press the Finish button. If the deployment was successfully you should be able to access your servlet via the following URL: If you wait 5 seconds and refresh the browser, the number should increase. You created your first working servlet with Eclipse WTP! Right-click on the project and select Export. Specify the target directory and press Finish. You can now import the WAR file to your production Tomcat system and test the web application.

Chapter 6 : Creating a Simple Web Application Using a MySQL Database - NetBeans IDE Tutorial

Section 1 Basics of a Web Application HTTP Request HTTP Response Section 2 Basic Request flow in a Java EE application Basic Servlet JSP with JSTL & EL MVC Pattern Basics Runnning in Tomcat.

Chapter 7 : Getting Started with Web Applications - The Java EE 5 Tutorial

In this tutorial, a Java web application communicates directly with a MySQL database using the Java Database Connectivity API. Essentially, it is the MySQL Connector/J JDBC Driver that enables communication between the Java code understood by the application server (the GlassFish server), and any content in SQL, the language understood by the.

Chapter 8 : Web Application basics Tutorial for Java beginners

Prerequisites for this Java web application tutorial. Before you begin this application development tutorial, you must have the following: If you don't have an Azure subscription, create a free account before you begin.

Chapter 9 : Struts Tutorial | Java Web Tutor

In this tutorial we'll create a "hello world" Java servlet. A servlet is a Java class that runs in an "application server" and sends web pages back to a browser when a user somewhere in the world clicks on a URL.