

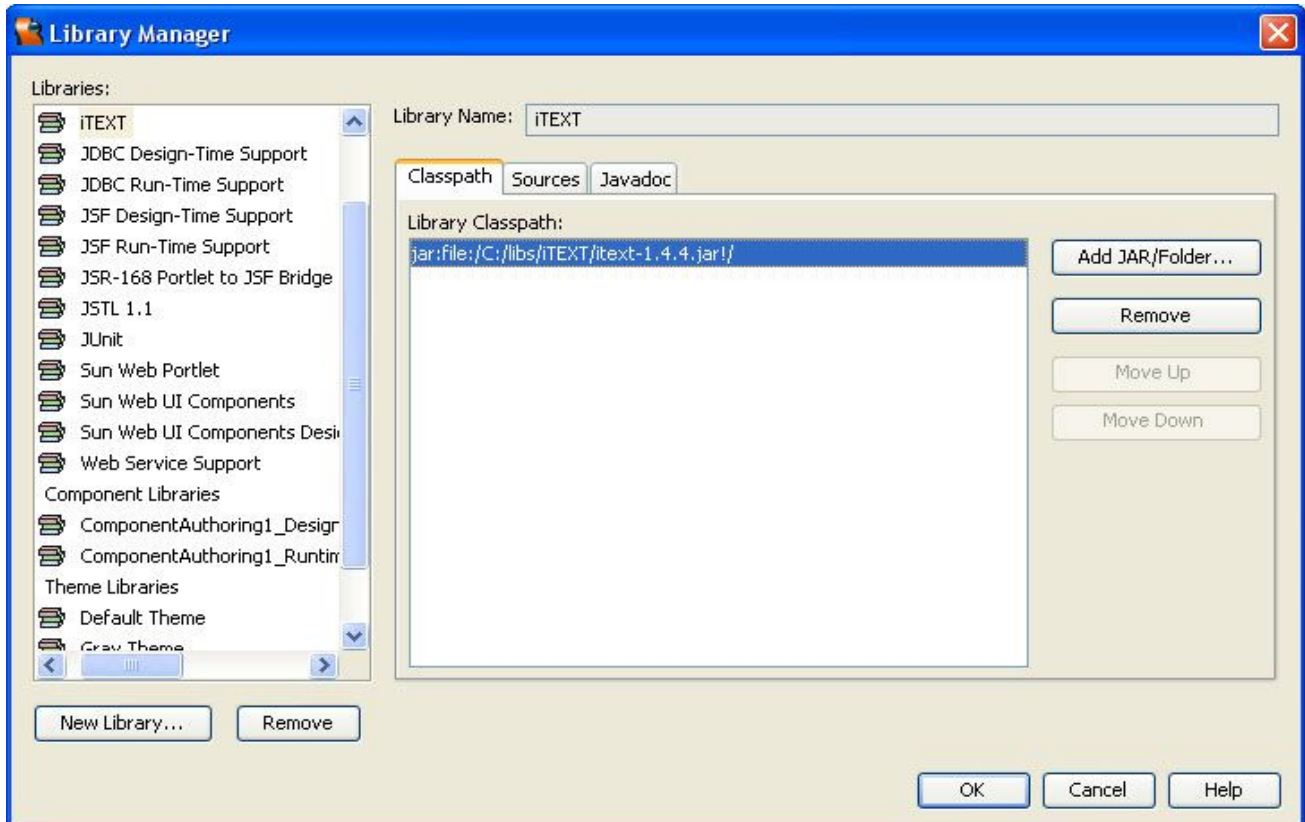
iText is a library that allows developers to extend the capabilities of their web server (and other JAVA) applications with dynamic PDF document generation.

Note: You might get a warning about rtf being deprecated. You can safely ignore this warning for now.

This tutorial will show you how to create simple PDF's using Sun Java Studio Creator and the iTEXT library.

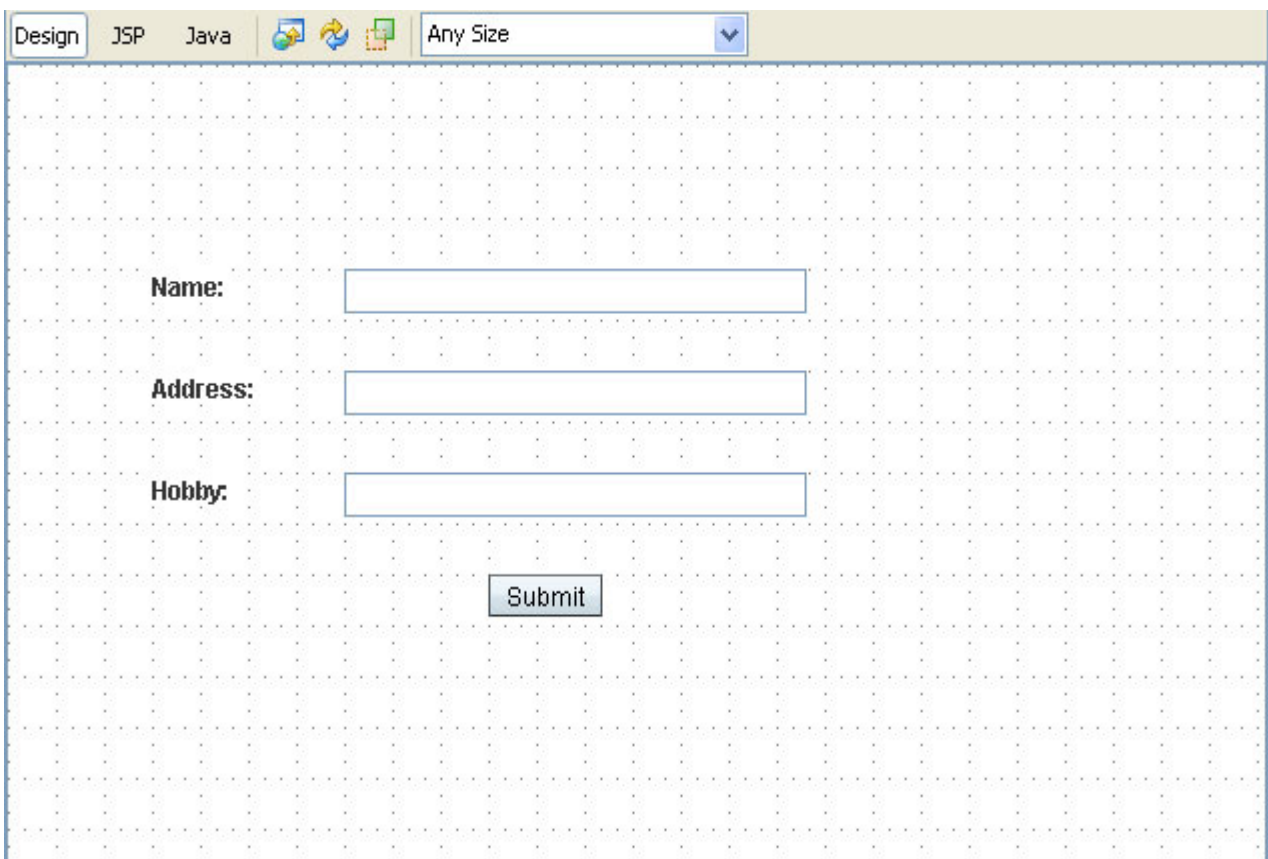
Creating a iTEXT Class Library

1. Download itext-1.4.4.jar and itext-docs-1.4.4.tar.gz from <http://www.lowagie.com/iText/>
2. In the IDE, choose Tools > Library Manager from the main menu
3. Click New Library, type iTEXT in the Library Name field, and click OK.
4. Click Add JAR/Folder and navigate to the directory where you saved itext-1.4.4.jar
5. To make the iTEXT Javadoc available to the Java Editor, select the Javadoc tab, click Add Zip/Folder, navigate to the directory into which you extracted the itext-docs-1.4.4.tar.gz files, select the directory, and press Enter.
6. Click OK to close the Library Manager.



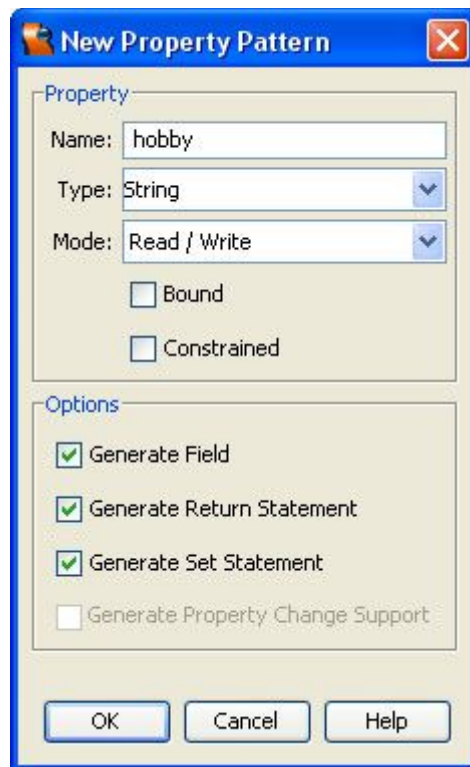
Creating our Main Page

1. Choose File > New Project, JSF Web Application and call it iTEXT-1
2. Add 3 Text Field Components, 3 Label Components and 1 button component to Page1. Change the text from the label components to: Name: Address: and Hobby: and attach the label components to the Text Field Components by selecting a label component, holding ctrl+shift and drag them to the Text Field Component. Change the text of the button component to Submit.

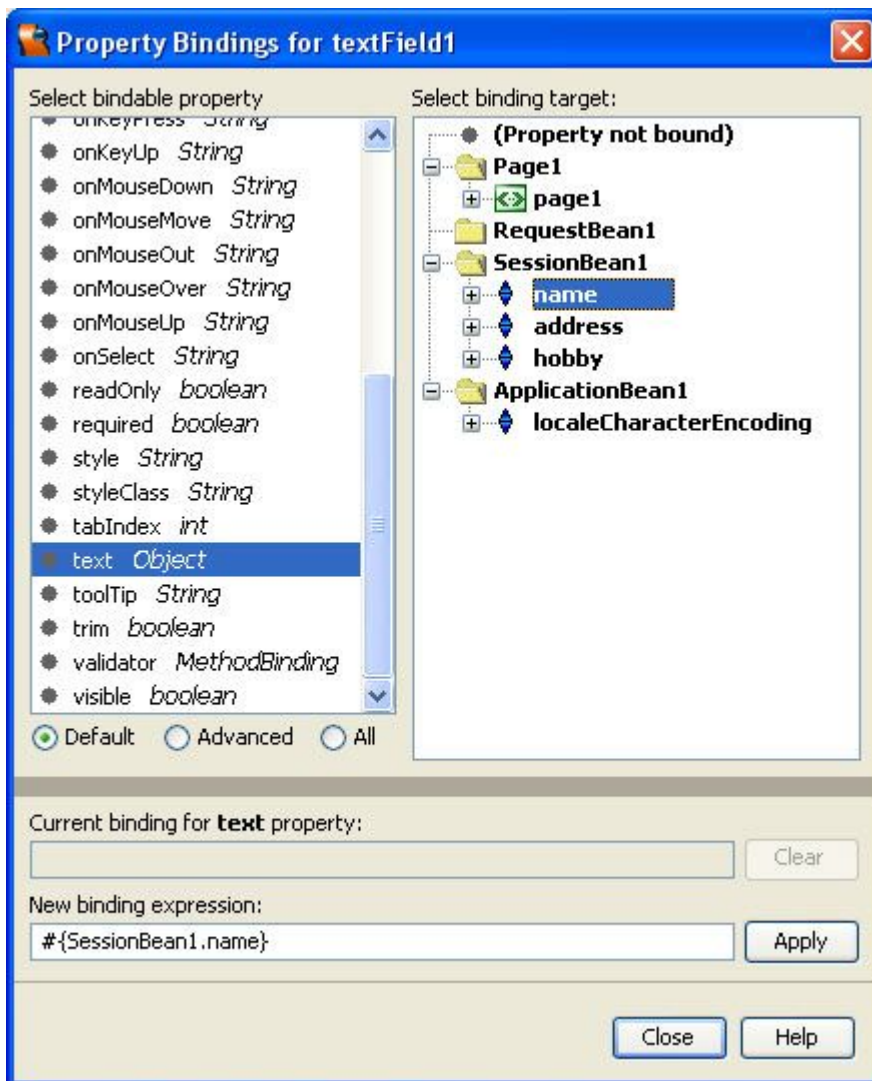


Storing the values inside SessionBean1

1. Create 3 SessionBean Property's by going to the project window, Expand Session Bean, Right click on SessionBean1, choose Add -> Property. Name the 3 properties as Name, Address, Hobby.

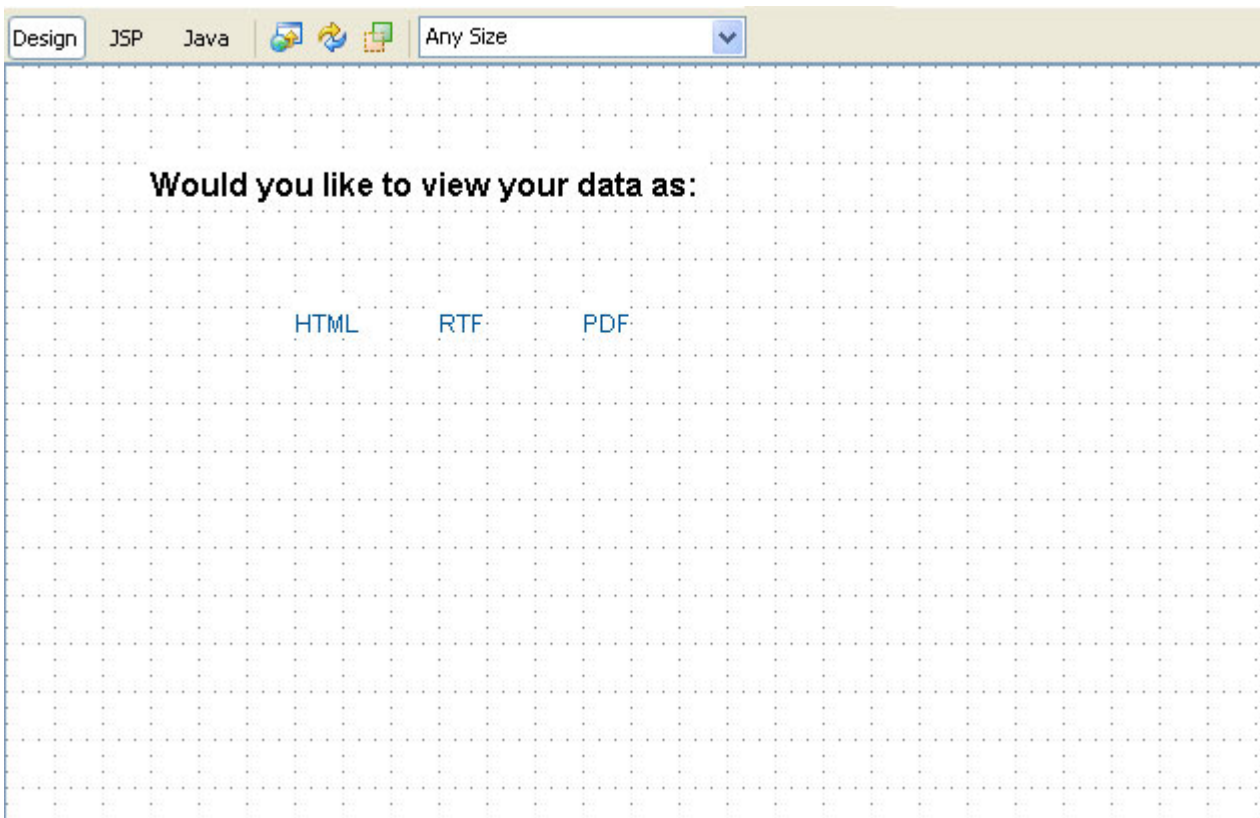


2. Now bind the Text Field Components together with the SessionBean properties. To do that right click on a Text Field Component -> Property Bindings and select the appropriate SessionBean property and next click on apply. Do this for all 3 Text Field Components.

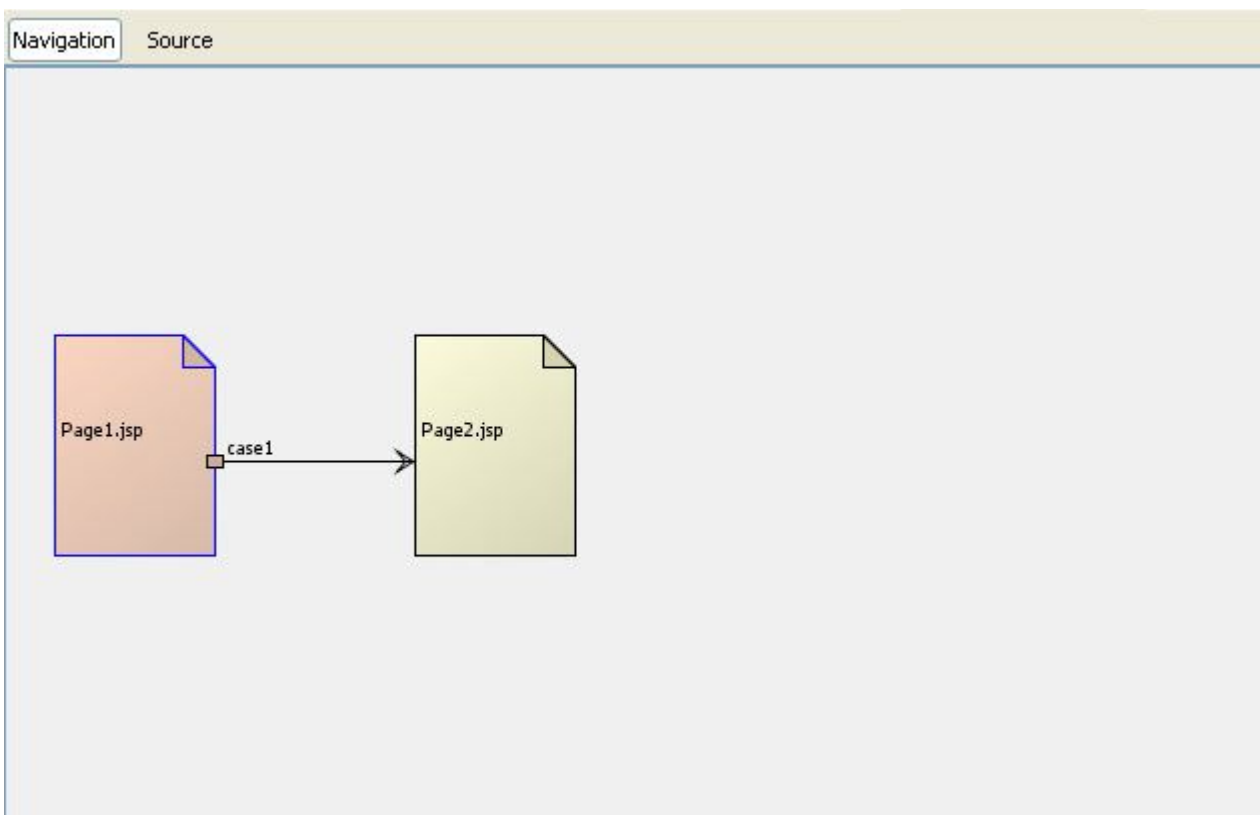


Create a Second Page with 3 output options

1. Create a new page, right click on Web Pages -> New -> Page , name is Page2 and press finished.
2. Drag a label component and 3 Hyperlink components on Page2. Change the text of the label component to: Would you like to view your data as:
 Change the text of your Hyperlink components to HTML, RTF and PDF. In the properties window of the Hyperlink components set the url to:
 servlet/iTEXT.pdf?presentationtype=html for HTML
 servlet/iTEXT.rtf?presentationtype=rtf for RTF
 servlet/iTEXT.pdf?presentationtype=pdf for PDF
 In the properties window for the label component set labelLevel to Strong



3. Right click inside Page2 and select Page Navigation. Drag the button component from Page1.jsp to Page2.jsp and press enter.



1. From the Projects Window right click on Libraries -> Add Library -> Select iTEXT -> Click Add Library.
2. Next from your Projects Window right click on Source Packages -> New -> Java Package. Fill in Package Name lowagie and click finish.
3. From the Projects Windows expand the Source Packages, Right Click on lowagie and choose New -> Java Class. Fill in Class Name servlet1 and choose finish.
4. Servlet1.java will be opened. Copy and paste the code below inside servlet1.java

```

package lowagie;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

import com.lowagie.text.Document;
import com.lowagie.text.DocumentException;
import com.lowagie.text.Paragraph;
import com.lowagie.text.html.HtmlWriter;
import com.lowagie.text.pdf.PdfWriter;
import com.lowagie.text.rtf.RtfWriter;
import itext1.SessionBean1;
import java.util.Date;

public class servlet1 extends HttpServlet {

    protected void doGet(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException {

        //Store SessionBean Properties inside variables
        itext1.SessionBean1 sessionBean = (SessionBean1)
request.getSession().getAttribute( "SessionBean1" );
        String name = sessionBean.getName();
        String address = sessionBean.getAddress();
        String hobby = sessionBean.getHobby();

        // we retrieve the presentationtype
        String presentationtype = request.getParameter("presentationtype");

        // step 1
        Document document = new Document();
        try {
            // step 2: we set the ContentType and create an instance of the
corresponding Writer
            if ("pdf".equals(presentationtype)) {
                response.setContentType("application/pdf");
                PdfWriter.getInstance(document, response.getOutputStream());
            } else if ("html".equals(presentationtype)) {
                response.setContentType("text/html");
                HtmlWriter.getInstance(document, response.getOutputStream());
            } else if ("rtf".equals(presentationtype)) {
                response.setContentType("text/rtf");
                RtfWriter.getInstance(document, response.getOutputStream());
            } else {
                response.sendRedirect("http://itextdocs.lowagie.com/tutorial/gen
eral/webapp/index.html#HelloWorld");
            }
        }
    }
}

```

```

        // step 3
        document.open();

        // step 4
        document.add(new Paragraph(name));
        document.add(new Paragraph(address));
        document.add(new Paragraph(hobby));
        document.add(new Paragraph(new Date().toString()));

    } catch(DocumentException de) {
        de.printStackTrace();
        System.err.println("document: " + de.getMessage());
    }

    // step 5: we close the document (the outputstream is also closed
internally)
    document.close();
}
public void doPost(HttpServletRequest request,
    HttpServletResponse response)
    throws ServletException, IOException {
    doGet(request, response);
}
}

```

Changing our descriptor file so we can access the servlet

1. From the files windows expand iTEXT-1 -> web -> WEB-INF -> right click web.xml and select edit. Switch to XML view in the top and add the following code:

```

<servlet>
    <servlet-name>Generate PDF or HTML</servlet-name>
    <servlet-class>lowagie.servlet1</servlet-class>
</servlet>
<servlet>
    <servlet-name>Generate RTF</servlet-name>
    <servlet-class>lowagie.servlet1</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>Generate PDF or HTML</servlet-name>
    <url-pattern>/servlet/iTEXT.pdf</url-pattern>
</servlet-mapping>
<servlet-mapping>
    <servlet-name>Generate RTF</servlet-name>
    <url-pattern>/servlet/iTEXT.rtf</url-pattern>
</servlet-mapping>

```

Testing your application.

Deploy the application and try to get the output in HTML, RTF and PDF.