Using Kelp with an IDE

Table of Contents

Chapter 1. Kelp

This document describes how to use Kelp to develop applications with XMLC for use with Enhydra 5.1. It assumes that you have a basic understanding of Enhydra and either Sun NetBeans for Java, Community Edition, Borland® or JBuilder.

Kelp is a set of tools that extend a Java integrated development environment (IDE) to simplify the development of applications for Enhydra 5.1

Kelp for NetBeans is a set of tools for Sun Netbeans for Java, Community Edition. For information on NetBeans for Java, visit the Sun NetBeans for Java website.

Kelp for JBuilder is a set of tools for Borland JBuilder. For information on JBuilder, visit the Borland website at: http://www.borland.com/jbuilder/ [http://www.borland.com/jbuilder/].

Note: Kelp for NetBeans and Kelp for JBuilder function very similarly. For the sake of clarity, this chapter is written for one IDE, NetBeans for Java, Community Edition. Any notable differences for Kelp for JBuilder are addressed in a separate section at the end of this chapter.

Kelp features

Kelp provides the following tools and features:

- Kelp Application Wizard

  The Kelp Application Wizard generates Web applications using either the Servlet API or the Enhydra programming model.

- XML Compiler integration

  The Kelp XMLC tool lets you set XMLC options, select markup language files (e.g., HTML, XML) to compile, and call XMLC from within the IDE to create classes that generate web content dynamically.

- Kelp Deployer

  The Kelp Deployer allows you to set up your project properties, copy static content to the document root, process templates, create deployable archives, and deploy the archives.

- Enhydra Import wizard

  The Import wizard allows you to import Enhydra projects that use GNU Makefiles into your IDE. The Import Wizard is currently supported in JBuilder only.

- DODS Generator

  Dods generator runs ant based xml files (build_dods.xml and build_java.xml) for generating sql and java files.

- Enhydra XMLC properties

  The XMLC properties give you full control over how XMLC builds Document Object Model (DOM) classes from your HTML files.

- Input Template property pages

  The Input Template property pages let you specify a list of strings to search and replace when you are generating files from templates.

- Build integration

  Through property pages, you can set up JBuilder to invoke XMLC and the Kelp Deployer whenever you make or rebuild your JBuilder project. This feature lets you quickly ensure that your files are updated without having to run the tools individually.

  Note: Build integration is not currently supported by Kelp in NetBeans. To build Enhydra projects with Kelp in NetBeans, you must invoke the Kelp XMLC tool before building the project, and the Kelp Deployer afterwards.

- Kelp sample projects

  These projects demonstrate techniques for creating dynamic Web pages with XMLC Web applications.

Chapter 2. Project development overview

This section outlines how Kelp can be used with an IDE to speed the development process. Using the Kelp tools and wizards together with those of the IDE, you can perform the following basic functions:

Generating or importing source files

Within NetBeans or JBuilder you can generate a new Web Application or Enhydra Application (super-servlet style application), using the Kelp Application Wizard. The generated framework of files and directories provide a useful starting point for developing applications and services for Enhydra 5.1.

If you have an Enhydra application, you can import the application into the IDE using the Enhydra Import Wizard. The Import Wizard sets XMLC properties from the settings in the makefiles. This saves you time and trouble by automating tasks such as setting up the mapping tables to customize generated class names.

Setting project properties

The Kelp project properties consist primarily of XMLC and deployment options. XMLC options can be used to set the name for a generated class file, save the generated Java source-code files, and more. The XMLC options can be specified for a file, a folder, or the project. Project settings will be overridden by folder settings, and folder settings will be overridden by file settings. For additional information about XMLC, refer to Chapter 5, "Enhydra XMLC," of the Developer's Guide.

Deployment properties specify what files to include in the deployable archive file and how to deploy the archive file.

Compiling projects

Once you have generated or added the source files to your project, you can compile the project to generate the necessary classes. If you are using XMLC in your project, you must run the Kelp XMLC tool before compiling the rest of your source files.

Note: Projects compiled or built from within the IDE do not use Ant. If you want to take advantage of some of the advanced Ant capabilities, such as the DODS tasks for automatically generating entity beans, you may want to forget building your project within the IDE.

Building and deploying archive files

After you have compiled your project you can use the Kelp Deployer to create and deploy an archive file. The Kelp Deployer only deploys to mapped drives.

Debugging projects

if you are having trouble running your application, you can use the debugging capabilities of the IDE to isolate the problem. NetBeans for Java supports remote debugging. JBuilder supports debugging for applications running within the IDE. Although the debugging capabilities of the supported IDEs differ, both methods are useful. For additional information refer to "Debugging Kelp projects" (html [using_kelp/debugging.html] , pdf [using_kelp/debugging.pdf]).

Chapter 3. Kelp topics

- Using the wizards and tools (html [using_kelp/tools.html] , pdf [using_kelp/tools.pdf])
- Setting properties (html [using_kelp/properties.html] , pdf [using_kelp/properties.pdf])
- Debugging Kelp projects (html [using_kelp/debugging.html] , pdf [using_kelp/debugging.pdf])

Chapter 4. Kelp sample projects

Kelp includes two sample projects that demonstrate how to use XMLC and Enhydra to create Web and wireless applications within JBuilder. If you are new to both Enhydra and Kelp, you can start learning Enhydra by running the Kelp sample application. The Kelp sample demonstrates only a small part of Enhydra's capabilities.

Once you feel comfortable with the Web application sample, examine the DiscRack example that comes with Enhydra, located in <Enhydra5.1_root>/examples/DiscRack. The DiscRack example provides a more complete application that incorporates JDBC access. You can run the DiscRack example using most JDBC-compliant data sources, including Oracle, Microsoft SQL Server, PostgreSQL, and InstantDB.

If you are already familiar with Enhydra, you can use DiscRack to see how to set up a project for your own applications. For more information on DiscRack, see Chapter 8, "DiscRack sample application," of Getting Started with Enhydra.

The sample servlets

The sample project consists of four presentation objects that show some of the common uses of XMLC. The sample is simplified in that it only contains a presentation layer. Production Web applications normally contain at least three packages, including presentation, business, and data. For a detailed explanation on how you can separate applications into these three functional areas, see Getting Started with Enhydra.

Note: If you have moved a sample project after running the Kelp Deployer, run the Kelp Deployer again before using the sample.

Each servlet dynamically generates HTML files using XMLC. The HTML source files are located in a resources directory. Each HTML file is compiled into a class file using XMLC. There is a corresponding Java file that implements HttpServlet for each HTML file. These files work with the generated HTML classes to create and process input from the Web pages. The Java files are located in the kelp.webapp.presentation package. The four servlets demonstrate the following:

- Greetings Servlet: This is similar to a traditional Hello World example. This servlet contains one HTML element that is set through XMLC to greeting the user with a phrase contained in the Java source.
- Table Servlet: One of the most common tasks in dynamic HTML generation is populating an HTML table. This servlet shows you how to define a table as a template in HTML and then populate it through Java when a user requests the page. In a real-world application, the data would most likely come from a JDBC data source. For the sake of simplicity, this example populates the table with an array of values that are hard coded into a Java file.
- New Node Servlet: XMLC allows you to insert HTML blocks from external sources into an existing page. This example shows a page containing a span of HTML that is read in from a text file. You can modify the text file to alter the page without recompiling the HTML or Java files.
- Form Servlet: You can use XMLC with HTML input fields to create data entry forms. This servlet shows you how to update a file on the server with input values retrieved from a Web browser. For simplicity, this example uses a property file to store displayed values. Normally, values are stored in a JDBC data source that is accessed through the business and data layers of an application.

Chapter 5. Kelp for JBuilder - differences

The primary functions of Kelp are the same regardless of the IDE used. However, there are some differences in how Kelp items are accessed from within JBuilder.

Using the Enhydra Import Wizard

The Enhydra Import wizard lets you quickly import source files from an existing Enhydra GNU Makefile project into a NetBeans or JBuilder project, capturing any XMLC options specified in the Makefile system.

As an example, this section describes how to import the DiscRack sample project to JBuilder.

Note: Applications generated by the Kelp Application Wizard from within the IDE do not use the Makefile system or Ant. Applications generated using the Kelp Application Wizard launched from the command line include a build.xml and are built with Ant.

The following steps explain how to import and run the DiscRack example using Kelp with JBuilder. The steps are divided into three sections:

- Importing DiscRack
- Building DiscRack
- Running DiscRack

If your application is running outside of JBuilder, you should be able to skip "Running DiscRack" .

Importing DiscRack

To import DiscRack into JBuilder, follow these steps:

- Build DiscRack as described in the README file. In addition to running XMLC and compiling the java source code, this generates the data package source code.
- Create a new JBuilder project file.

  - In JBuilder, select File|New Project.

    This opens the Project Wizard dialog box.

  - In the Project Wizard dialog box, set Project Name to DiscRack.

    This name has no impact on the project or application, but we recommend that you use DiscRack for consistency.

  - Set the Root Path to <Enhydra5.1_root>/examples, where <Enhydra5.1_root> is the directory where Enhydra is installed.

    For example, the Root Path might be C:/Enhydra5.1/examples. You can click the button to the right of the field to navigate to the directory.

  - In the Project Directory Name field, enter DiscRack.
  - Accept the defaults for the rest of the fields.

    - Click Finish to generate the new project.
- Choose Wizards|Enhydra Import.
- Set the project directory to <Enhydra5.1_root>/examples/DiscRack.
- Click Next to navigate through the wizard, accepting all default settings.
- Click Finish to import the project.

Building DiscRack

To build DiscRack, follow these steps:

- Open the XML Compiler.
- Click Compile to generate the DOM Java source files using XMLC.
- When XMLC is finished compiling, click Close.
- Choose Project|Make to compile the Java source files.
- Choose Tools |Kelp Deployer.
- Click Deploy to copy static content files, process configuration templates, and create a deployable archive.

Running DiscRack

To run DiscRack, follow these steps:

- Choose Project|Project Properties.
- Add the InstantDB library to the required library list:

    - Select the Paths tab.
    - Select the Required Libraries subtab.
    - Click Add.
    - Click New.
    - Name the library "InstantDB".
    - Click Add.
    - Navigate to the location of idb.jar, select it, and click OK.
    - Click OK until the Project Properties dialog box is closed.
- Choose Run|Run Project.
- Open a browser to http://localhost:5555 where 5555 is the specified port, to view the application.