Contents

Preface v

1 Introduction 1

1.1 What is Scout? . . . 1

1.1.1 End User Perspective . . . 1

1.1.2 Management Perspective . . . 4

1.1.3 Developer Perspective . . . 6

1.2 Why Scout? . . . 7

1.3 What should I read? . . . 9

1.3.1 I know Java . . . 9

1.3.2 I know tons of both Java and Eclipse . . . 10

1.3.3 I am a manager . . . 10

2 "Hello World" Tutorial 11

2.1 Installation and Setup . . . 11

2.2 Create a new Project . . . 11

2.3 Run the Initial Application . . . 14

2.4 The User Interface Part . . . 15

2.5 The Server Part . . . 18

3.2.1 Desktop 29
3.2.2 Form . . . . . .

4.4.12 Creating new Application Modules 69
4.4.13 The NLS Editor 69

5 A Larger Example 73
5.1 The "My Contacts" Application 73
5.2 Setting up the new Scout project 78
5.3 Adding the Person Page 81
5.4 Adding the Company Page 84
5.5 Installing the Databas .ee 73

Preface

Today, the Java platform is widely seen as the primary choice for implementing enterprise applications. While many successful frameworks support the development of persistence layers and business services, implementing front-ends in a simple and clean way remains a challenge. This is

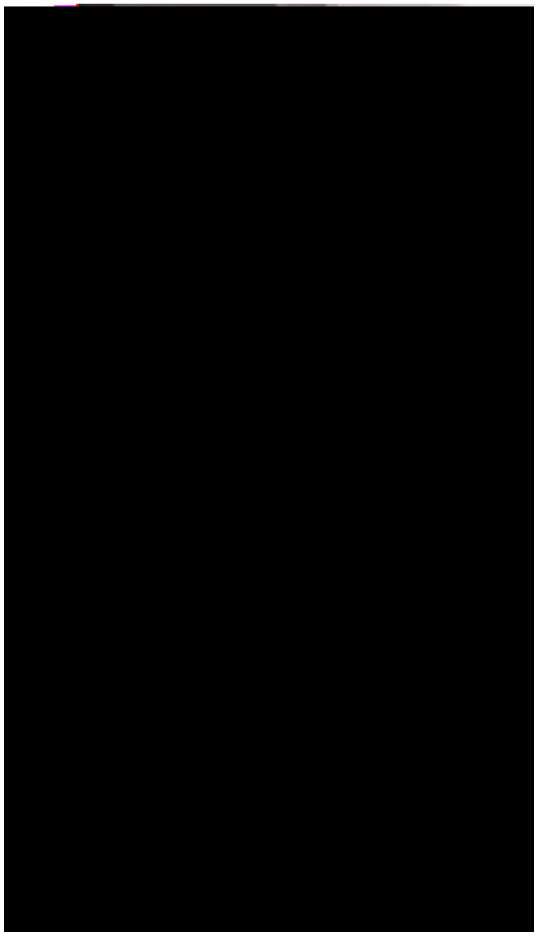


Figure 1.5: The integration of a Scout application in a typical enterprise setup.

'lightweight' framework is frequently developed. When available, this framework initially leads to desirable gains in productivity. Unfortunately, such frameworks often become legacy by themselves.

developer productivity and helps to motivate the development team. Additional reasons on why

Finally, Scout is an open source framework hosted at the Eclipse foundation. This provides a number of interesting options to developers that are not available for closed source frameworks. First of all, it is simple to get all the source code of Scout and the underlying Eclipse platform. This allows for complete debugging of all problems and errors found in Scout applications. Starting from the application code, including the Scout framework, Eclipse and down to the Java platform.

Scout developer can also pro t from an increasing amount of free and publicly available documentation, such as this book or the Scout Wiki pages. And 1(co Td7l)-1(dm)-397(twith-397(Scout)-398(Wr)-3
Thiou-3917al-3918ageast-3918alatcelopereky-3917asitutionTohe Idellyoresetinnsvi28(or)-3426arm-3

widely adopted by in the industries and unlikely to become legacy in the foreseeable future. While for the back-end side of enterprise applications well-known and proven frameworks do exist, the situation on the client side is less clear. Unfortunately, user interface (UI) technologies often have lifetimes that are substantially shorter than the lifetimes of larger mission critical applications. This is particularly true for the web, where many of today's frameworks will no longer be relevant in ve or more years.

Enter Eclipse Scout. This open source framework covers most of the recurring needs that are relevant to the front-end development of business applications. And Scout forces a clean separation between the user interface and the speci c UI technology used for rendering. This has two major

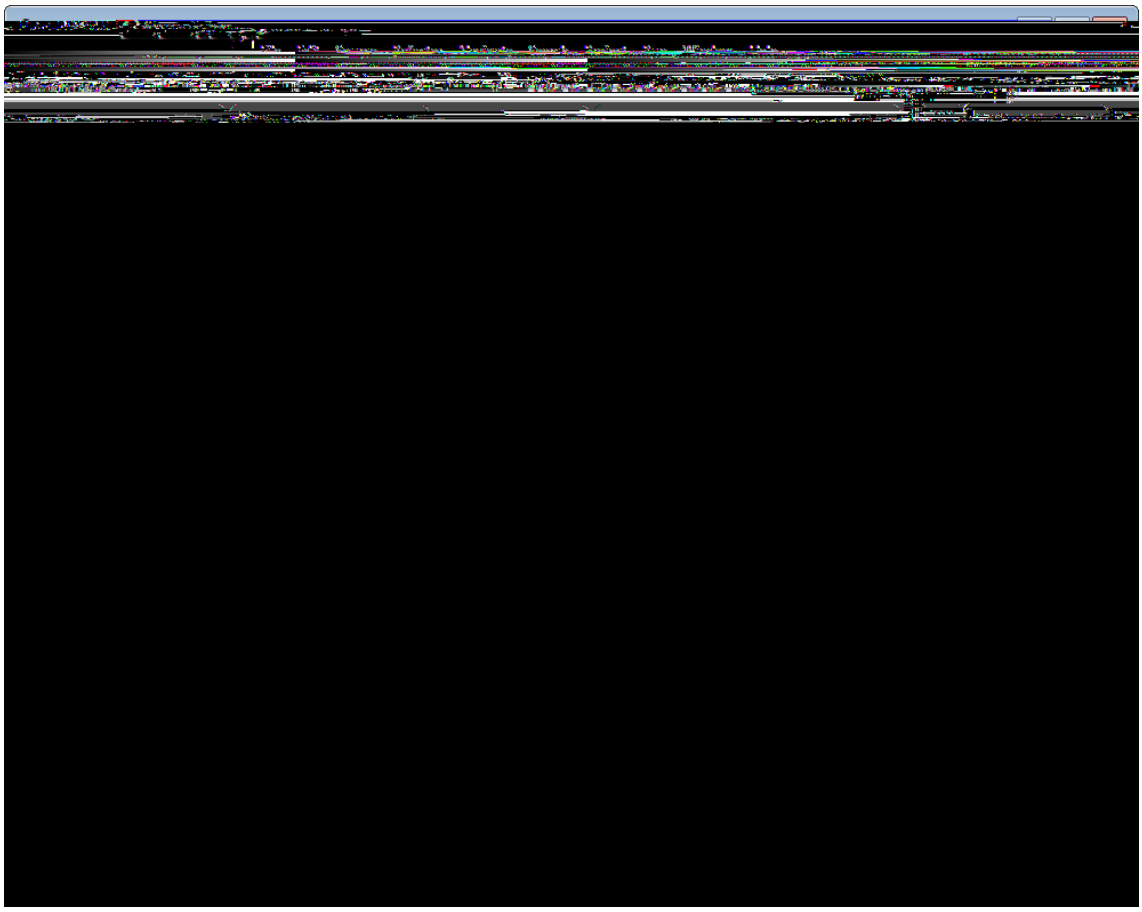
1.3.2 I know tons of both Java and Eclipse

This means that you are one of these software wizards that get easily bored. You prefer to get a quick impression before deciding to dig deeper and hate going through lengthy descriptions. In

Figure 2.1: Create a new Scout project using the Scout SDK perspective.

Figure 2.2: The new Scout project wizard.

In the *New Scout Project* wizard enter a name for your Scout project. As we are creating a "Hello World" application, use `org.eclipsescout.helloworld` for the *Project Name* eld according to Figure 2.2. Then, click the Finish button to let the Scout SDK create the initial project code for you.



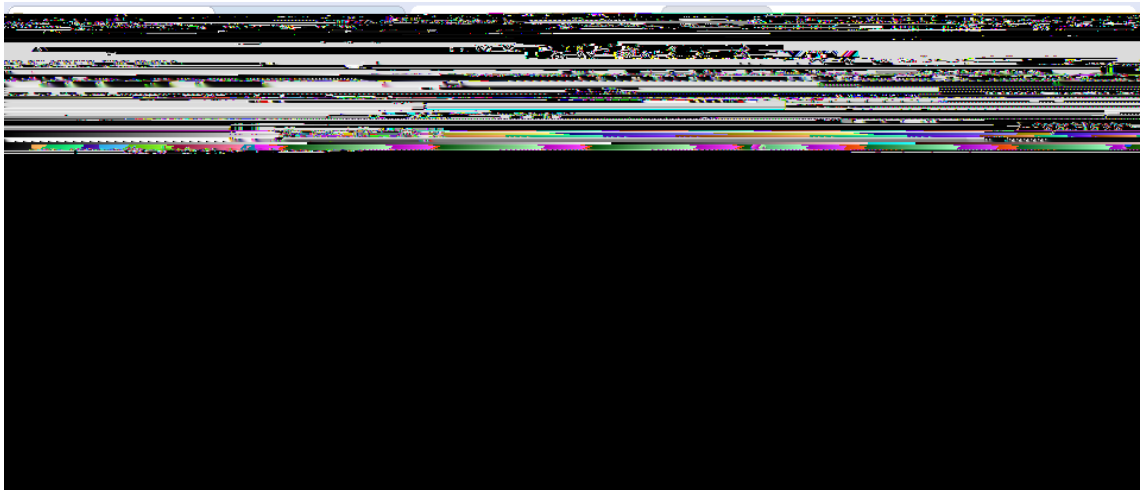
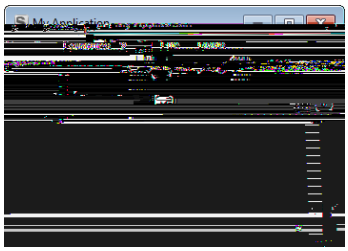
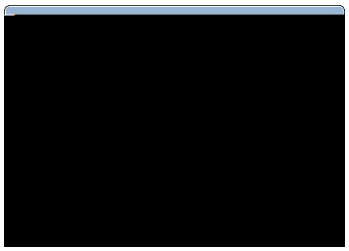


Figure 2.4: Starting the web client in the Scout SDK using the provided RAP product launcher. Make sure to start the server before starting any client product.

2.3 Run the Initial Application

After the initial project creation step we are ready to start the server and the clients of the still empty Scout application. For this, we switch to the Scout Explorer and select the root node *org.eclipse.scout.helloworld*





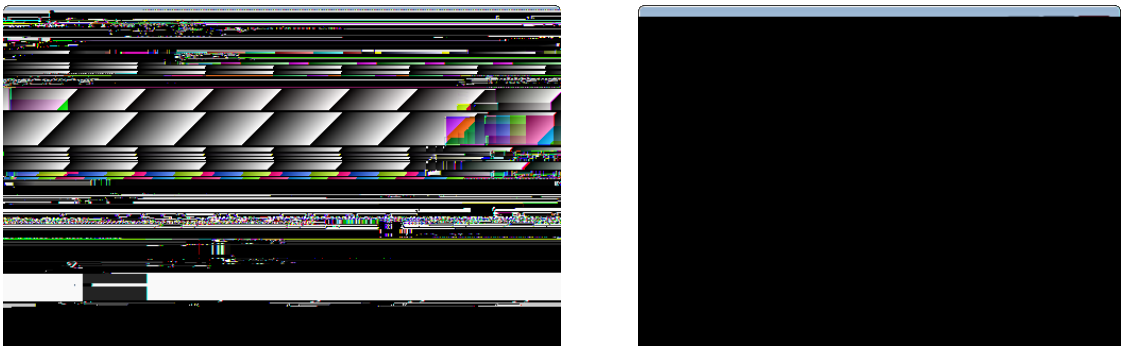


Figure 2.7: Adding the *DesktopBox* eld with the Scout SDK form eld wizard.

Figure 2.9: Adding a new translation entry.

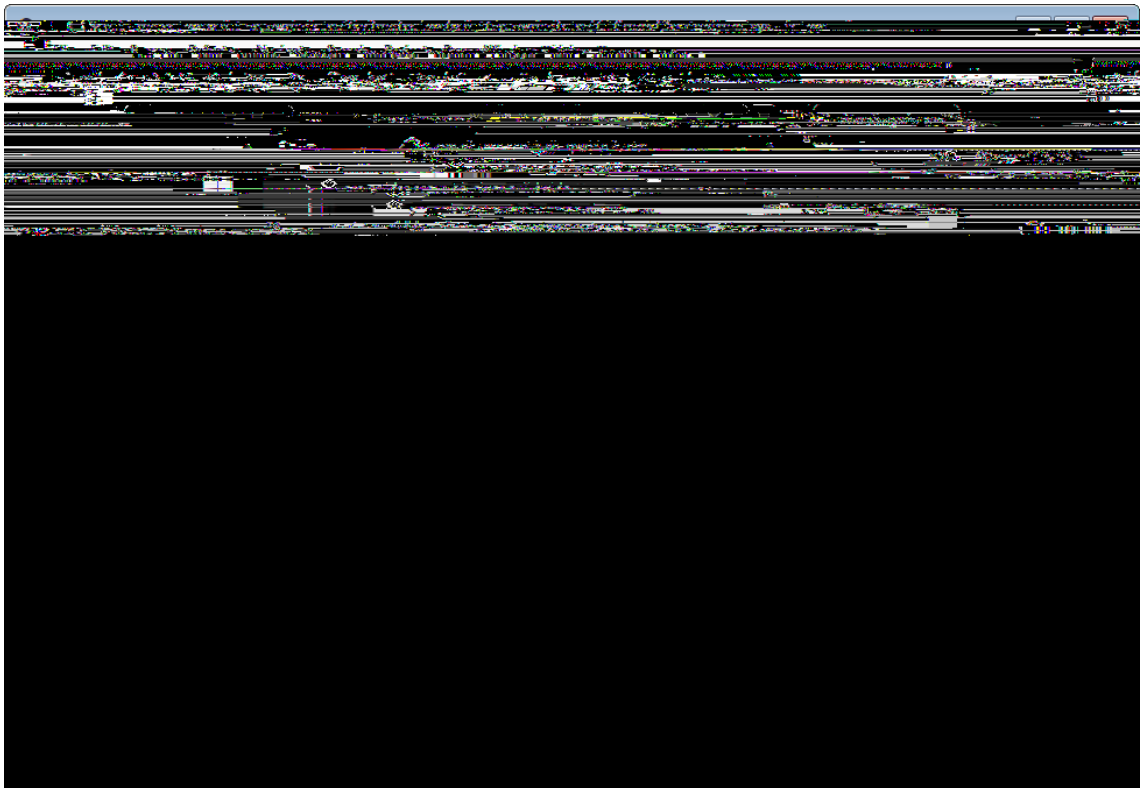


Figure 2.10: Scout SDK showing the *MessageField*

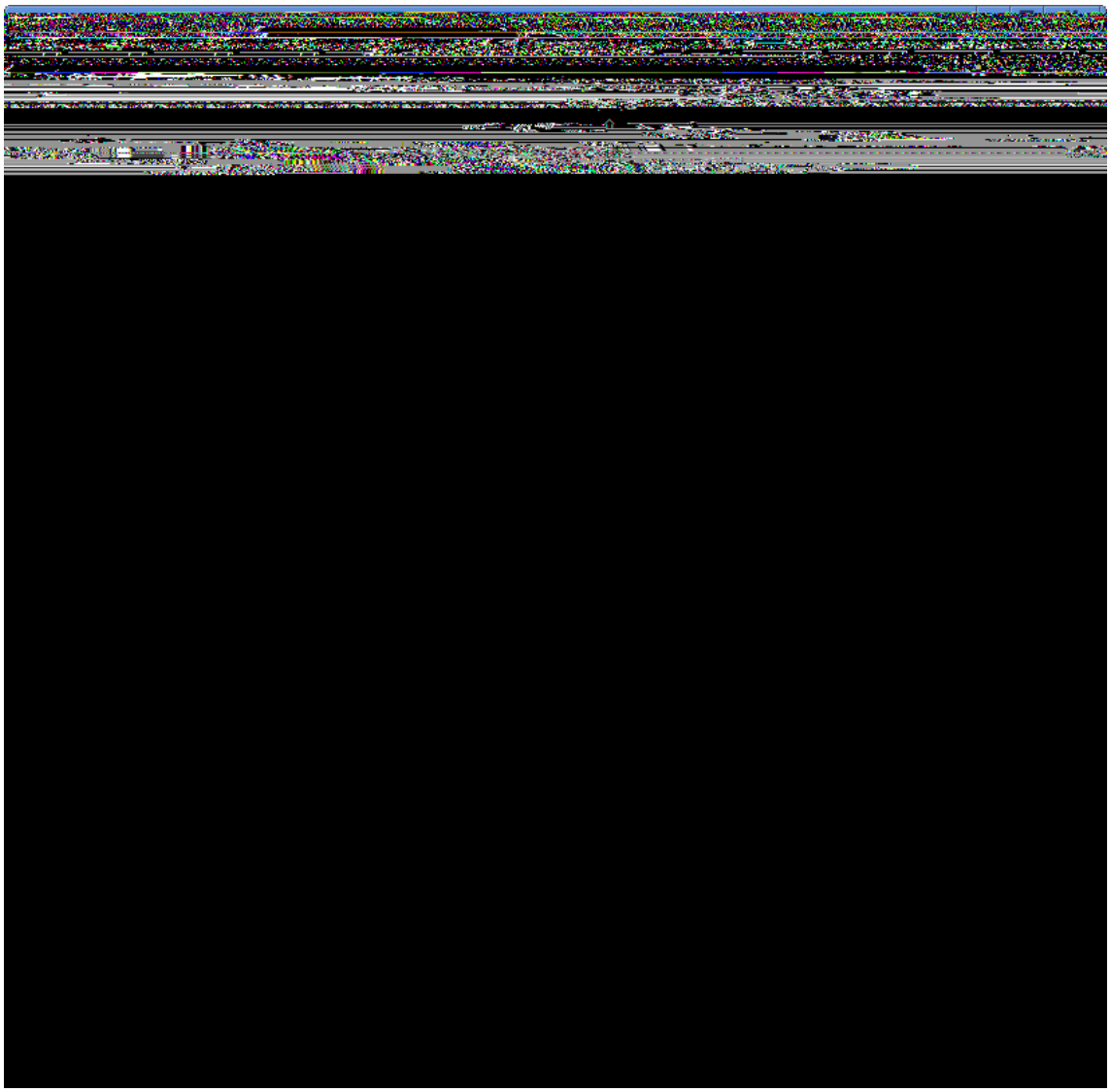


Figure 2.19: The "Tomcat Web Application Manager". The WAR files to be deployed can then be selected using button "Choose File" highlighted in red.

Chapter 3

"Hello World" Background

Listing 3.2: Class `DesktopForm` with its view handler and `startView` method. Other inner classes and methods are omitted here.

```
public class DesktopForm extends AbstractForm {
  public class ViewHandler extends AbstractFormHandler {

    @Override
    protected void execLoad() throws ProcessingException {
      IDesktopService service = SERVICES.getService(IDesktopService.
          class);
      DesktopFormData formData = new DesktopFormData();
      exportFormData(formData);
      formData = service.load(formData);
      importFormData(formData);

    }
  }

  public void startView() throws ProcessingException {
    startInternal(new ViewHandler());
  }
```

Figure 3.3: Using the Edit Content... icon shown on the left hand side, the product selection

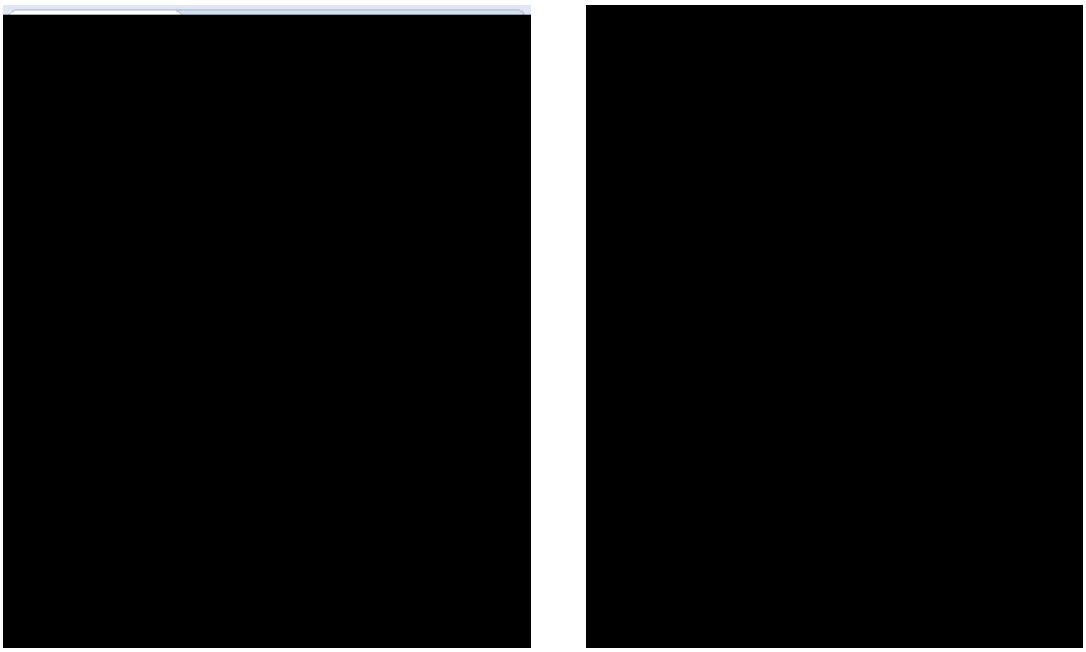


Figure 3.4:

Listing 3.3: The DesktopForm with its inner class MainBox containing the desktop box and message field

```
@FormData(value = DesktopFormData.class, sdkCommand = FormData.
    SdkCommand.CREATE)
public class DesktopForm extends AbstractForm {
  @Order(10.0)
  public class MainBox extends AbstractGroupBox {
```

```
DesktopForm
        extends
```

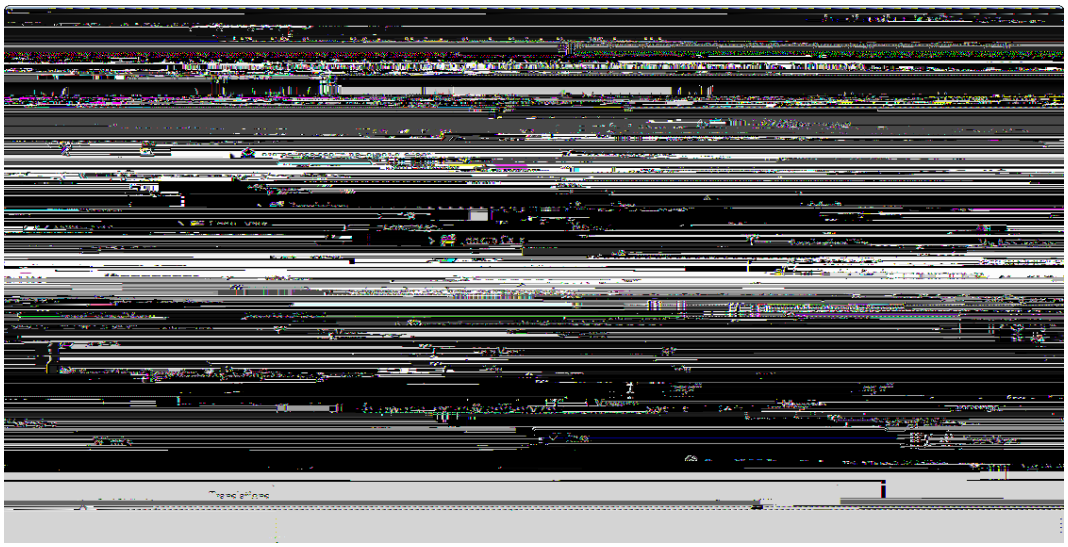


Figure 3.7: The NLS editor provided by the Scout SDK. This editor is opened via the *Open NLS Editor ...* link in the Scout Object Properties of the *HelloworldTextProviderService* node.

Listing 3.5: The server service class `DesktopService`.

```
public class DesktopService extends AbstractService implements
```

Listing 3.7: The registration of the `IDesktopService` proxy service in the client plugin of the "Hello World" application. This is the complete content of the client's `plugin.xml` file.

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin>

  <extension
       name=""
       point="org.eclipse.scout.service.services">
```

```
="org.eclipse.scout.
```

Chapter 4

Scout Tooling

Figure 4.1:

4.2 The Scout Explorer

The Scout Explorer view is responsible for the navigation support within the Scout application model. This navigation support is presented in the form of a tree view and includes the client

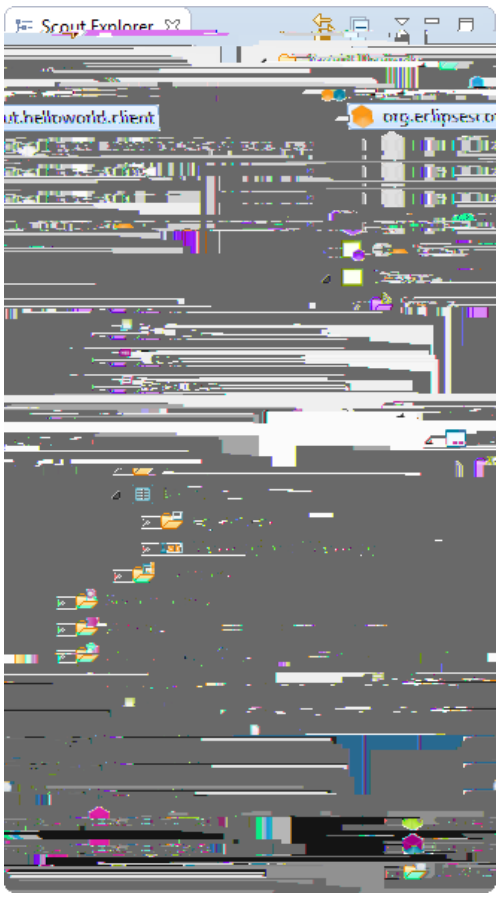
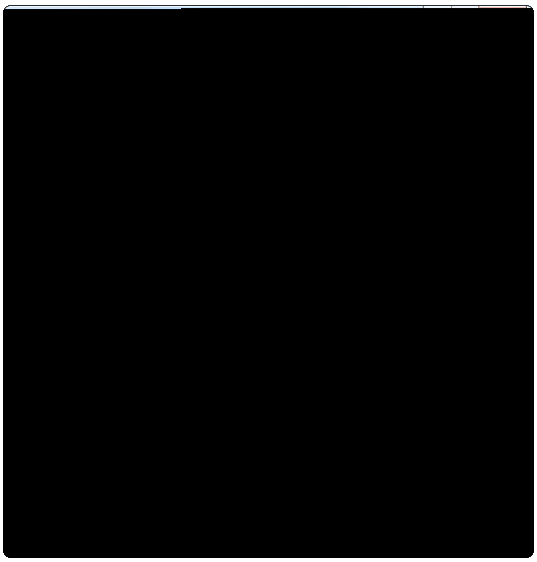


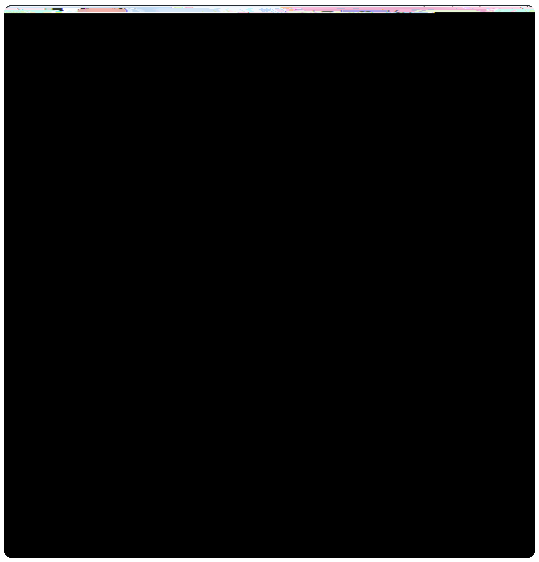
Figure 4.2: The Scout Explorer view. The grey nodes below the expanded client node represent the supported U:7273 07Hchnologies.

Figure 4.3: Thecout with7273 07he expanded node.

Figure 4.8: The last wizard step is used to specify the RAP target for the new Scout application.

Figure 4.10:





6Figure33944.

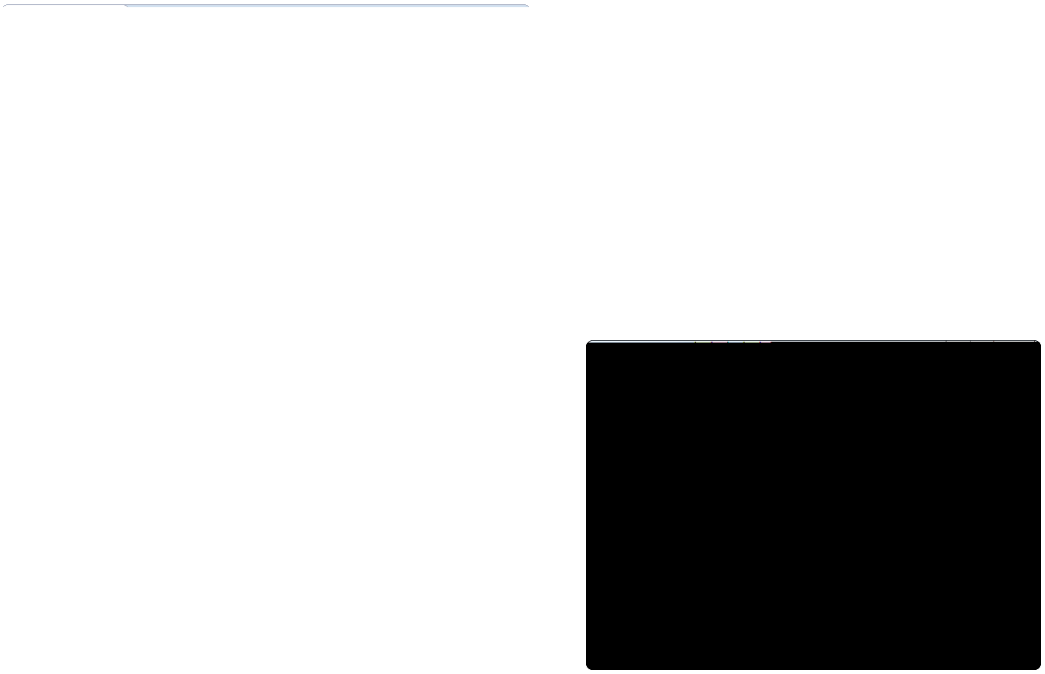


Figure 4.13: Starting the new form field wizard with the context menu on a composite field (left). The first wizard step (right) is used to select the field type.

Figure 4.17: The Scout SDK wizard to create a new page with table.

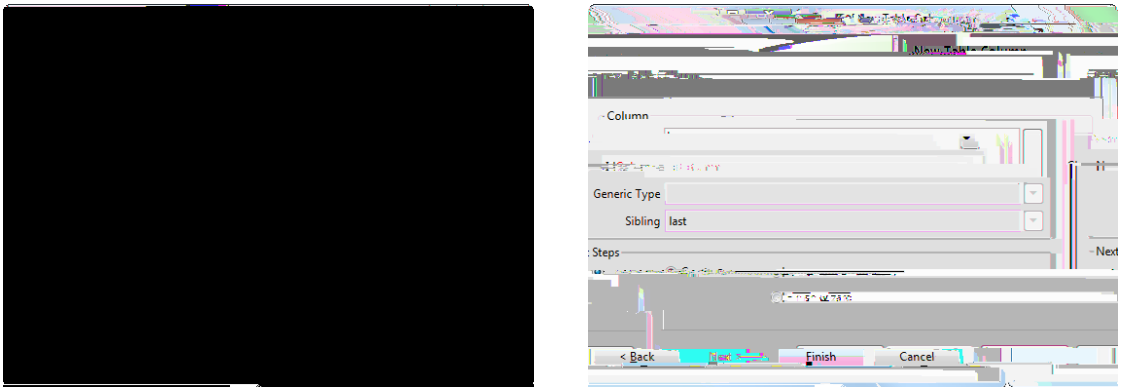


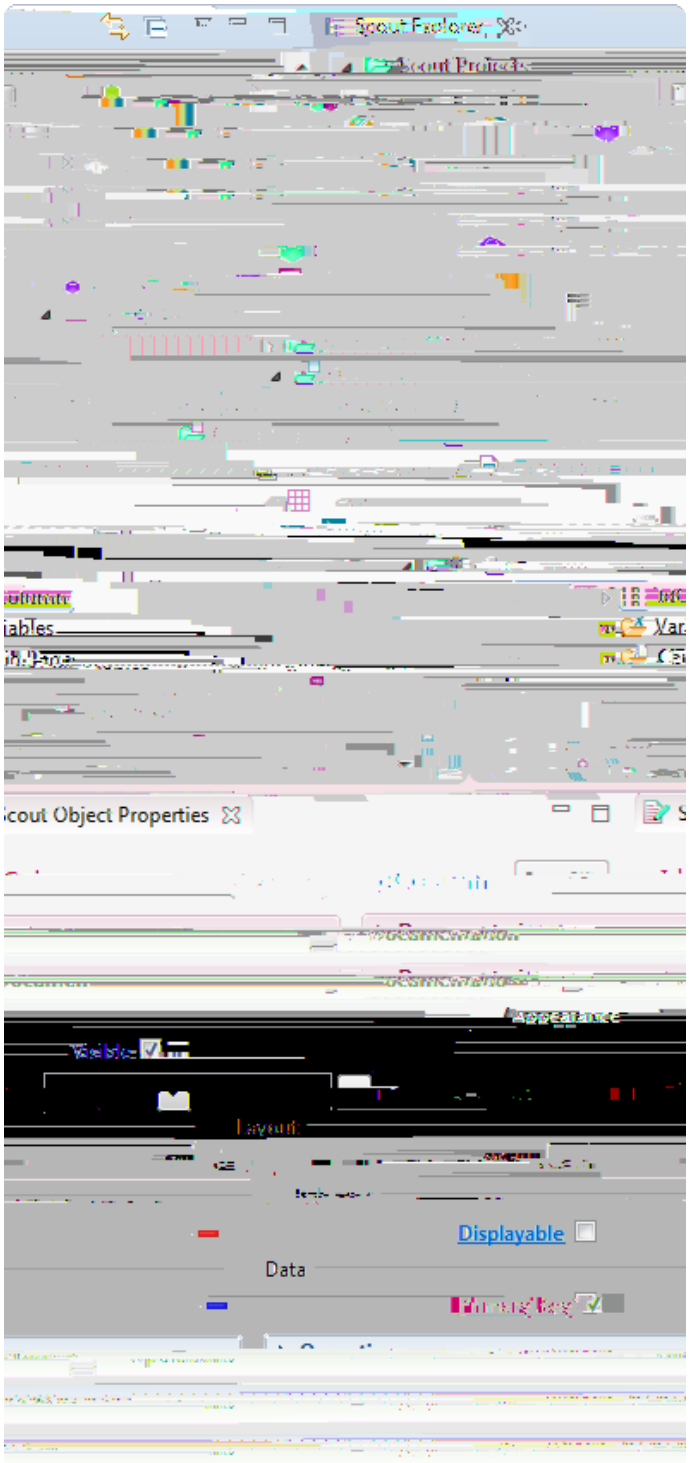
Figure 4.18: The Scout SDK wizard to create a new table column.

4.4.7 Creating new Table Columns

The *New Column* wizard allows to create columns in a table. In the Scout Explorer, the New Column... context menu is available below tables, on the *Columns* folder. The *New Column* wizard is shown in Figure 4.18.

4.4.8 Creating new Search Forms

The *New Search Form* wizard allows to create a new Scout form to search for elements displayed in pages. In the Scout Explorer, the Create Search Form... context menu is provided on the *Search Forms*


Scout Explorer
Scout Project
Scout Object Properties
Appearance
Visible
Layout
Displayable
Data

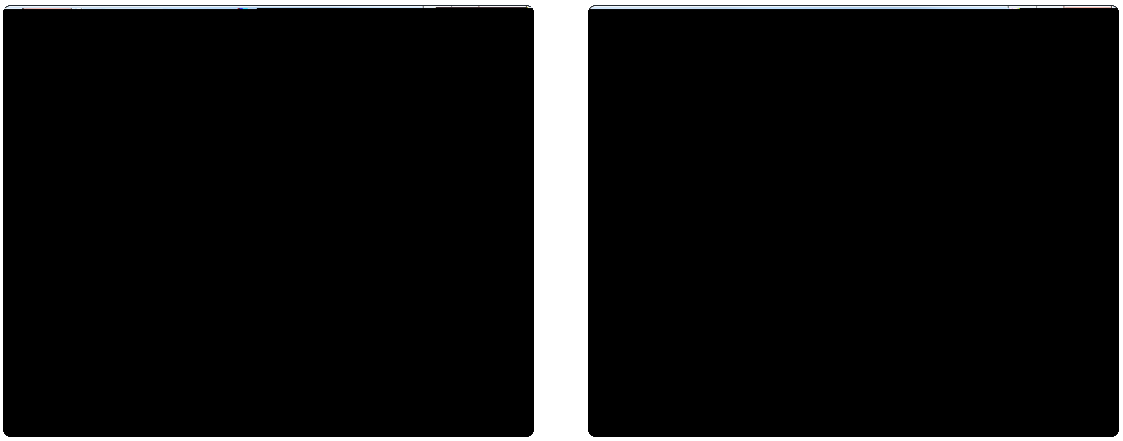


Figure 4.22: The Scout SDK wizard to create a new service.

Figure 4.23: The Scout SDK wizards to create a new code type (left) and a new code (right).

4.4.10 Creating new Code Types and Codes

The *New Code Type* wizard allows to create new code types. As access to code types and codes is required from both client and server applications they are located in the application's shared

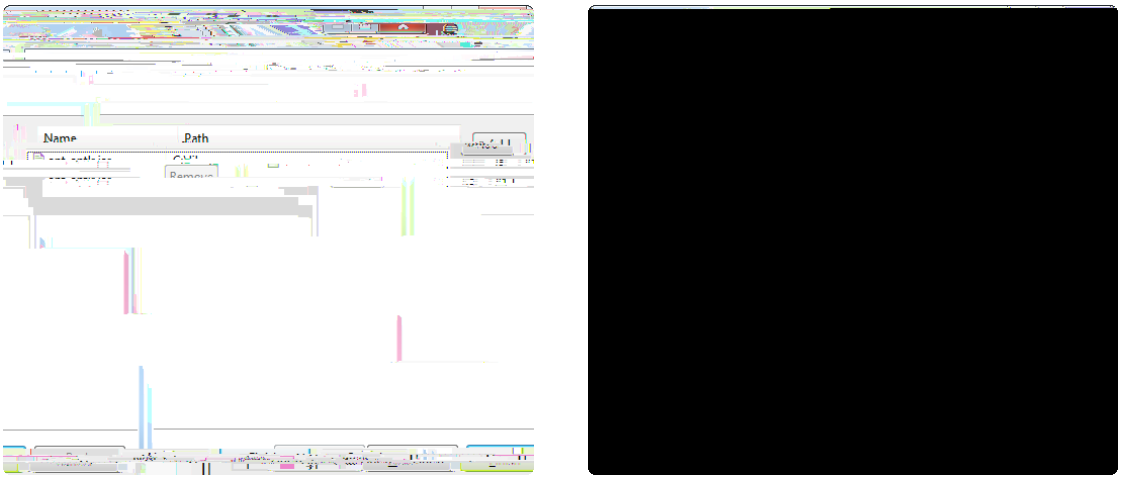


Figure 4.24: The Scout SDK wizard to add external JAR files in the form of a library bundle to a Scout project.

4.4.11 Creating new Library Bundles

The *New Library Bundle* wizard allows to add functionality provided in standard JAR files to a Scout application. In the Scout Explorer, the New Library Bundle... context menu is provided on the top-level *Libraries*

Figure 4.25: The Scout SDK wizard to create a new application module.

Figure 4.26: The NLS editor provided by the Scout SDK. This editor is opened via the *Open NLS Editor ...* link in the Scout Object Properties of the *DefaultTextProviderService* node.

Chapter 5

A Larger Example

In this chapter we will create the "My Contacts" Scout application. This small[1] application covers additional aspects of the Eclipse Scout framework. The presented demo application borrows heavily from a Scout tutorial published 2012 in the German *Java Magazin*[2] for the Scout release 3.8 (Juno). Compared to the 2012 tutorial, the version presented in this chapter has been slightly polished and updated to the Scout release 4.0 (Luna).

According to this step-by-step tutorial, we will build an outline based Scout application featuring a navigation tree and pages to present information in tabular form. In addition, the application also shows how to work with forms to enter and/or update data, menus and context menus. On the server side, we show how to work with databases, how to use logging in Scout applications and how to include standard Java libraries in the form of JAR files.

The chapter is organized as follows. In the first section, the finished demo application is

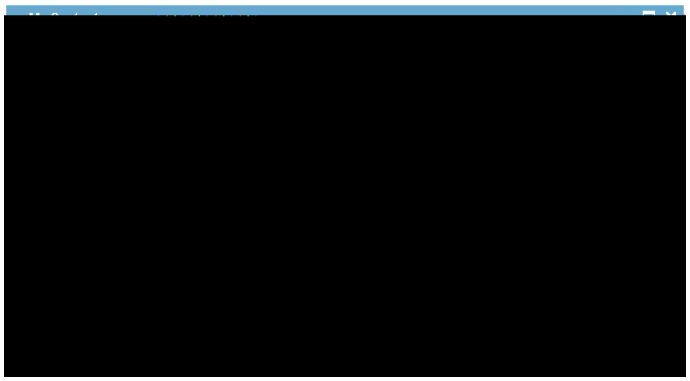
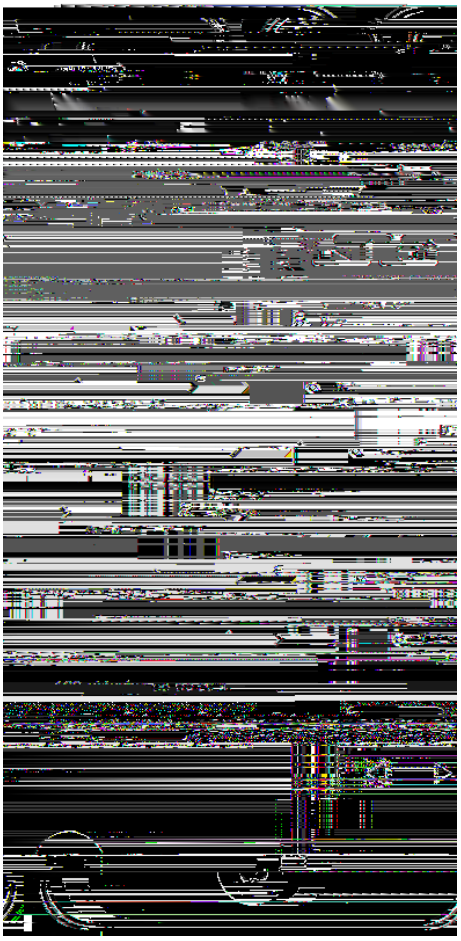


Figure 5.2:

Figure 5.7: The "My Contacts" application running in the browser as web application. The Excel



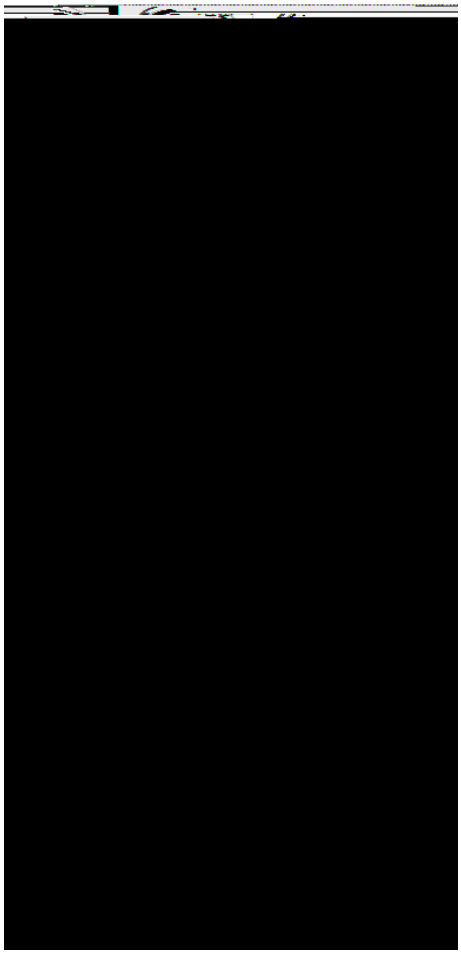


Figure 5.8:

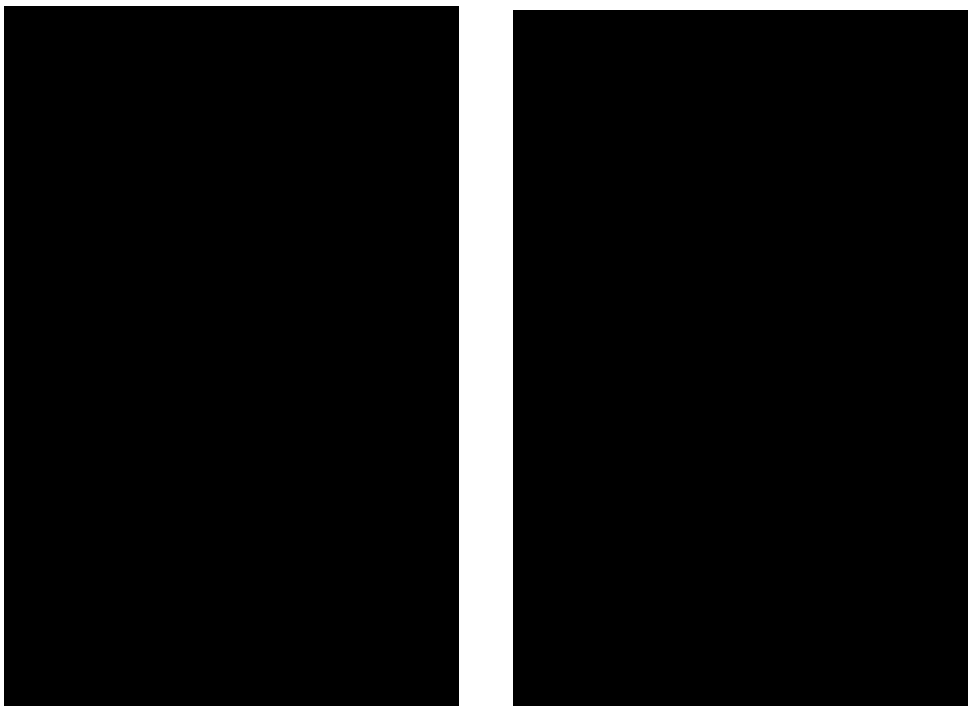
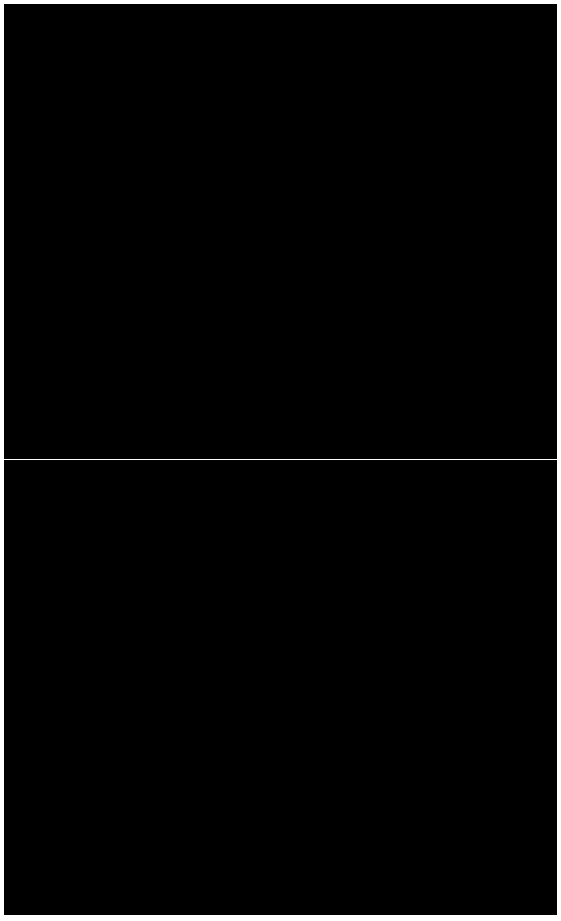


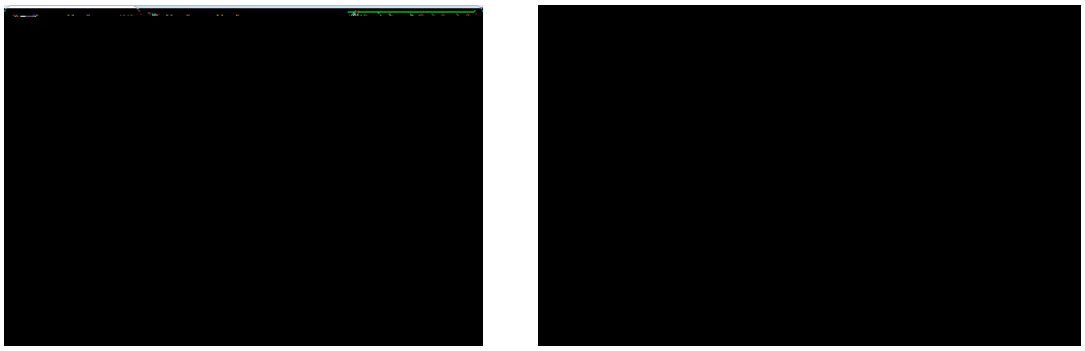
Figure 5.11: Con gure the application name "My Contacts" in the title eld of the Desktop's properties.

Remark: Adding the Docx4j support will add the Export to Excel menu under tools menu on the application's desktop. This presence of this feature will then allow to export contacts shown in the application to an Excel sheet. As shown in Listing 5.1, the ScoutXlsxSpreadsheetAdapter rst creates an excel le based on the currently active page in the



Listing 5.3: The `execCreateChildPages`

Figure 5.19: Add a database installation service.



Listing 5.8: The `installStorage` service operation to create the database schema for the "My

Listing 5.9: Setting up the COMPANY table of the "My Contacts" application.

```
tprivate void createCompanyTable(String
```

The second feature is the parameter binding used in the `INSERT INTO` statements. When SQLEg 0 243.692 -1el(tsEg 0 2are)- 0 2executed any 43.6ic

Listing 5.12: Scheduling the database installation in the `start` method of the `ServerApplication` class during the server application startup.

```
public class ServerApplication implements IApplication {
  private static IScoutLogger logger = ScoutLogManager.getLogger(
    ServerApplication.class);

  @Override
  public
```

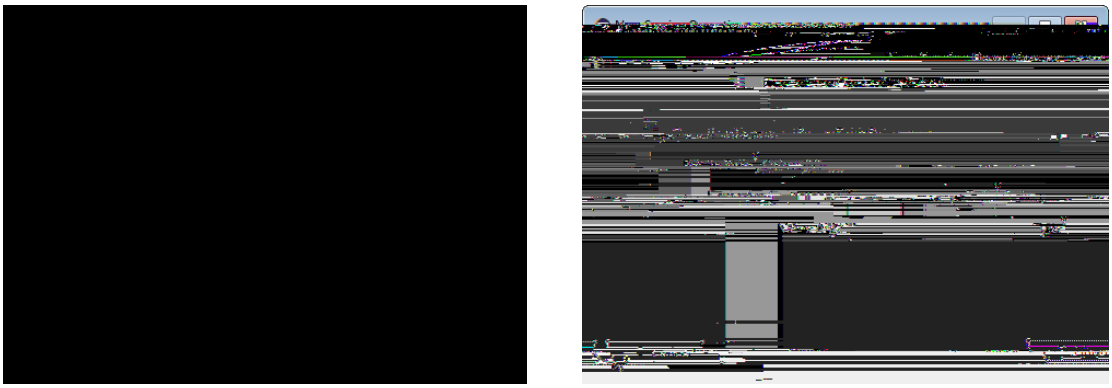
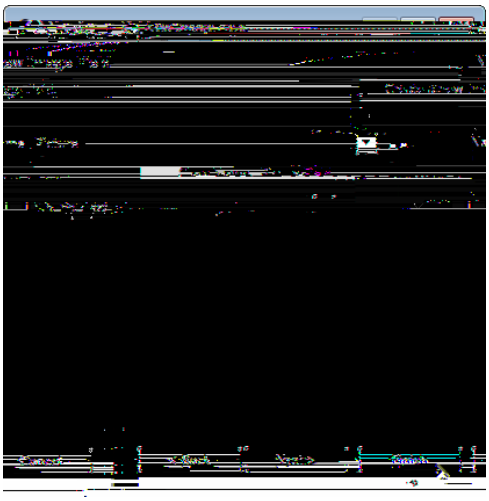


Figure 5.21: Add the service operation to fetch the data for the person table.

shown in Figure 5.21. In the wizard dialog enter "getPersonTableData" into the *Operation Name* eld. For the return type we simply use a two dimensional object array. As the person page is also displayed under the company page, we need to have a way to only return the persons working for a speci c company. To allow for this use case, we add the parameter "companyId" as the rst argument to the `getPersonTableData` operation before we close the wizard with Finish button.

The next operation we need is the `getCompanyTableData` method. You may use the same creation steps as described above for the person table page data. But345(pat.TJ/F42(fetc)28(kint345(("compan)27(y

Listing 5.14:



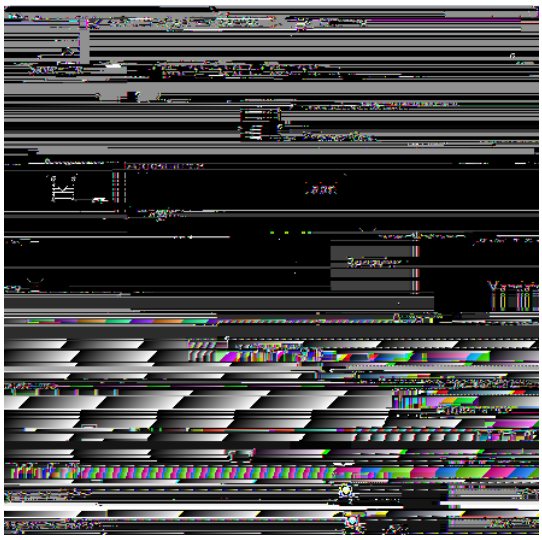


Figure 5.28:

Listing 5.16:

Listing 5.18: The edit menu implemented in class EditURLMenu of the picture field. If the URL was changed the picture URL field of the person form is set accordingly in method execAction

```
@Order(10.0)
public class EditURLMenu extends AbstractExtensibleMenu {

  @Override
  protected String getConfiguredText() {
    return TEXTS.get("EditURL");
  }

  @Override
  protected void execAction() throws ProcessingException {
    PictureURLForm form = new PictureURLForm();
    form.startModify();
    form.waitFor();
    if 308Td(R8waitis(Stoext);)]T {
```

exec&0a0G0.50.970.97ra0.970.970.97RG0a0G0.970.970.97ra0.970.970.97RGETa1001100.346628.508.8100425.19

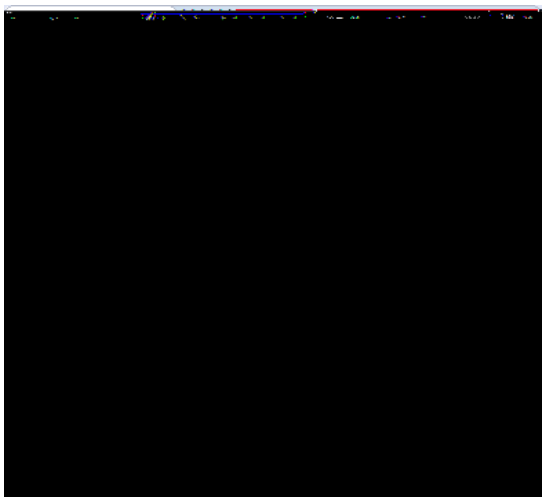


Figure 5.32:

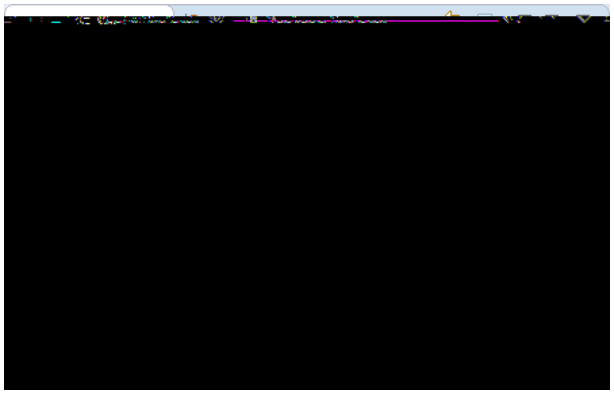
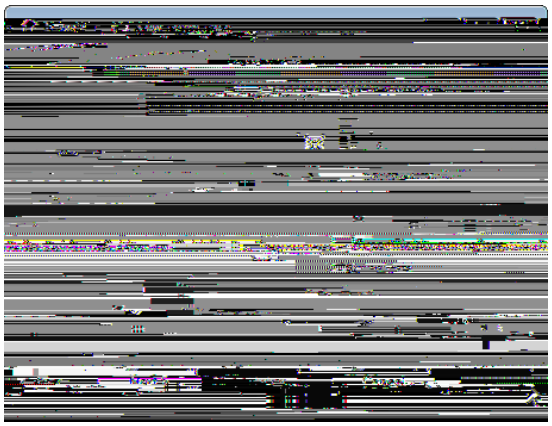
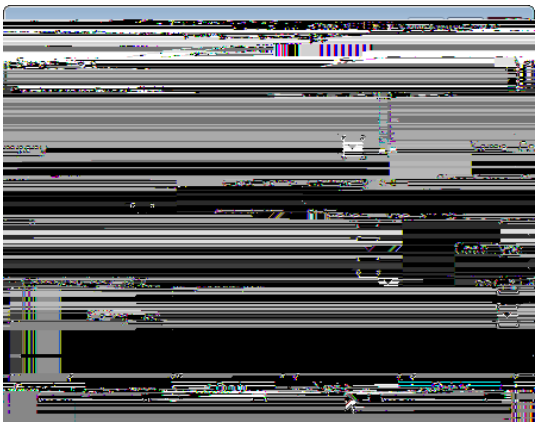


Figure 5.33:

Listing 5.22: The company lookup service in the application's server plugin. The `key` and the `text` criteria are used to search for values by key or by the provided name substring. index

```
public class
```





Listing 5.23: The smart field `CompanyField` of the person form and its wiring with the company lookup call.

```
@Order(40.0)
public class CompanyField extends AbstractSmartField<String> {

  @Override
  protected String getConfiguredLabel() {
    return TEXTS.get("Company");
  }

  @Override
```

Listing 5.27: The `load` and the `store` methods of the server's `PersonService`.

```
public class PersonService extends AbstractService implements
  IPersonService {
 @Override
 public PersonFormData load(PersonFormData formData) throws
   ProcessingException {
  if (!ACCESS.check(new ReadPersonPermission())) {
   throw new VetoException(TEXTS.get("AuthorizationFailed"));
  }

  SQL.selectInto("SELECT "
     + "picture_url, "
     + "first_name, "
     + "last_name, "
     + "company_id, "
     + "headline, "
     + "location, "
     + "date_of_birth "
     + "FROM
```

Listing 5.28: The

Figure 5.37: Add a new bundle to hold the Scribe JARs.

`companyId` variable to the company form. To decide on the elds that need to be on the company form you may check the setup of the database schema provided in Listing 5.9. If in doubt about what to do, please refer to the procedure used to create the person form.

In case youou73 7535e a2ohe database

Figure 5.38: Specify the JARs to be contained in the library bundle and the library name.

The Scout SDK wizard then creates the corresponding library plugin and updates the server product files and the plugin dependencies of the applications server plugin accordingly. Once the wizard has completed the classes defined in the two JAR files can directly be accessed from the "My Contact" application's server plugin.

5.11 Integrating LinkedIn Access with Scribe

Method `refreshToken` can now be implemented according to Listing 5.31. Using the provided token parameters and the security code, the access token is created using the `m_`

Listing 5.32: The `readContacts` method to fetch the users connection using the LinkedIn API. The necessary access token is created in method `getToken` based on the information stored in the CHAed 5.104.1G0593.98d //5.32: A578.446 cm680 425.1he 11(ed5 re fQ4 8.9664)the CHA4PTER 5.104.1G0582.032 if5.32:

Listing 5.34: A sample contact represented in the XML format provided by the LinkedIn API.

Listing 5.35: The `DomUtility` class provides functions to parse the XML data structure provided

Listing 5.36: The `updateContacts` method is used to enter/update existing contacts based on newG/mho on

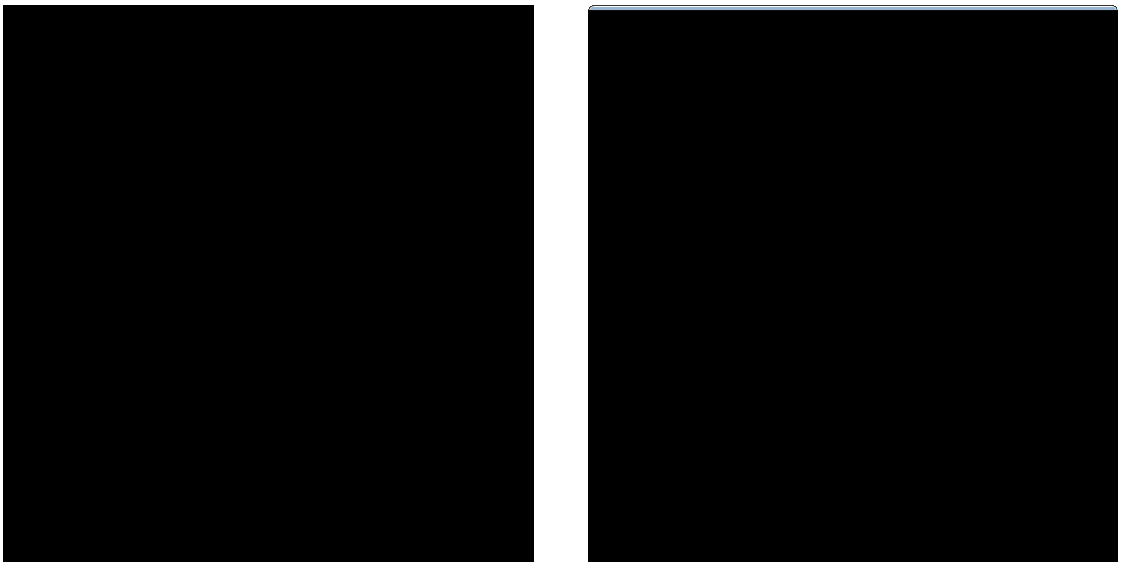
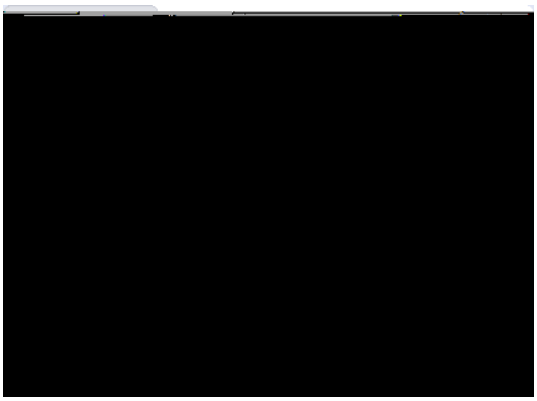
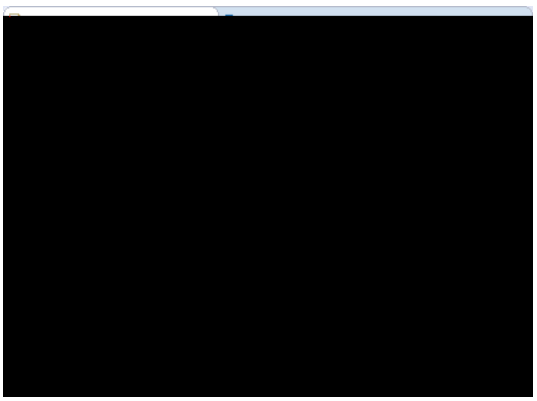


Figure 5.42: Add the form to refresh the LinkedIn access token.

vice the person's attributes are loaded from the server's database. The LinkedIn attributes are then used to update the corresponding form data. With

C Listing 33055.





Appendix A

Licence and Copyright

This appendix first provides a summary of the Creative Commons (CC-BY) licence used for this book. The licence is followed by the complete list of the contributing individuals, and the full licence text.

A.1 Licence Summary

This work is licensed under the Creative Commons Attribution License. To view a copy of this

Your fair dealing or fair use rights, or other applicable copyright exceptions and limitations;

The author's moral rights;

Rights other persons may have either in the work itself or in how the work is used, such as publicity or privacy rights.

Notice

f. "Work" means the literary and/or artistic work offered under the terms of this License including without limitation any production

3. License Grant. Subject to the terms and conditions of this License, Licensor hereby grants You a worldwide, royalty-free, non-exclusive, perpetual (for the duration of the applicable copyright) license to exercise the rights in the Work as stated below:

   a. to Reproduce the Work, to incorporate the Work into one or more Collections, and to Reproduce the Work as incorporated in the Collections;

   b. to create and Reproduce Adaptations provided that any such Adaptation, including any translation in any medium, takes reasonable steps to clearly label, demarcate or otherwise identify that changes were made to the original Work. For examplnon-exclder600(translation)-60coulded rkidedthe originalWork

      the originalWorkas difped."ns;

       inCollectionse a,s;

Original Author"). The credit required by this Section 4 (b) may be implemented in any reasonable manner; provided, however, that in the case of a Adaptation or Collection, at a minimum such credit will appear, if a credit for all contributing authors of the Adaptation or Collection appears, then as part of these credits and in a manner at least as prominent as the credits for the other contributing authors. For the avoidance of doubt, You may only use the credit required by this Section for the purpose of attribution in the manner set out above and, by exercising

WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. Termination

   a. This License and the rights granted hereunder will terminate automatically upon any breach by You of the terms of this License. Individuals or entities who have received Adaptations or Collections from You under this License, however, will not

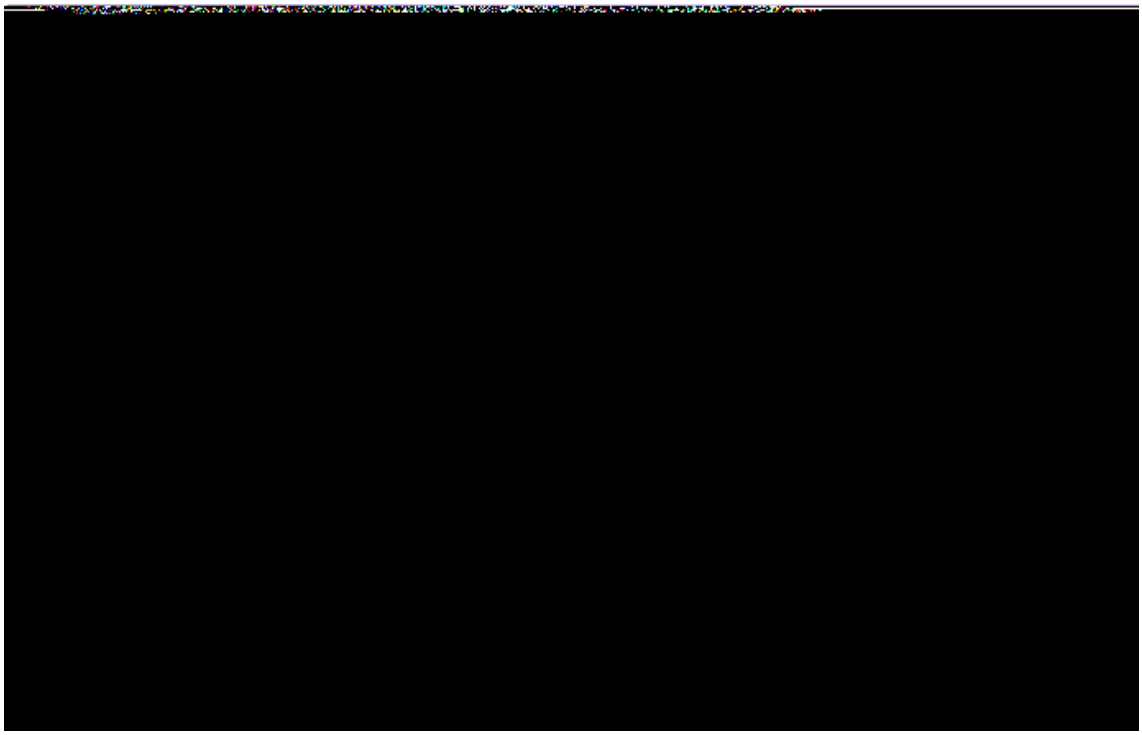
Appendix B

Scout Installation

B.1 Overview

This chapter walks you through the installation of Eclipse Scout. The installation description (as well as the rest of this book) is written and tested for Eclipse Scout 4.0 which is delivered as integral

Figure B.1:



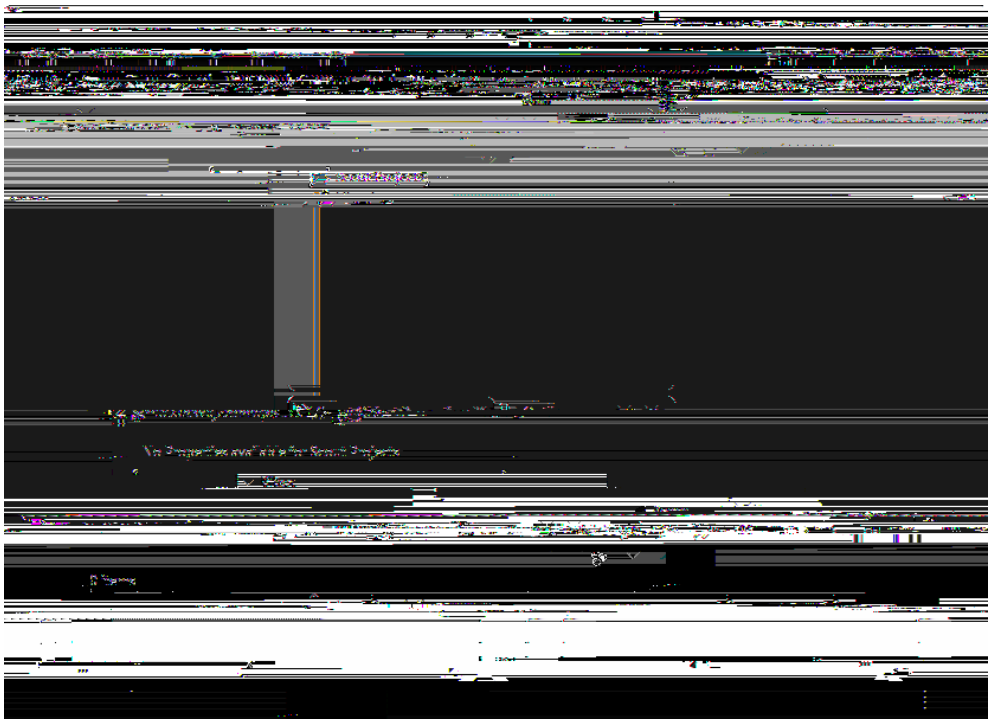


Figure B.6: Eclipse Scout started in the Scout SDK perspective.

Figure B.8:

B.5 Verifying the Installation

After you can start your Eclipse Scout package you need to verify that Scout is working as intended. The simplest way to verify your Scout installation is to create a "Hello World" Scout project and run the corresponding Scout application as described in Chapter 2.

Appendix C

Apache Tomcat Installation

Apache Tomcat is an open source web server that is a widely used implementation of the Java Servlet Specification. Specifically, Tomcat works very well to run the server part of Scout client server applications. In case you are interested in getting some general context around Tomcat you could start with the Wikipedia article[1]

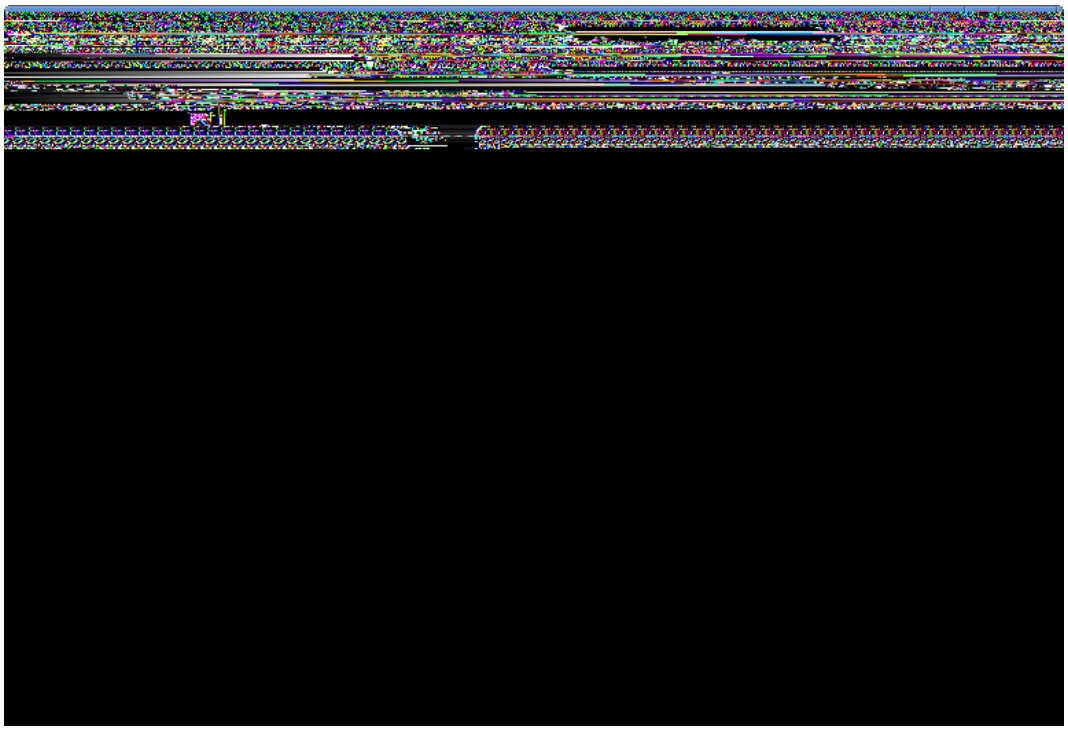


Figure C.1: A successful Tomcat 7 installation.

C.2 Directories and Files

Tomcat's installation directory follows the same organisation on all platforms. Here, we will only