End-to-end Rich Client Platform Solutions

Tools and Techniques
Who am I?

- David Orme
  - Senior Engineer for db4objects

- Open-source project leadership:
  - Eclipse Visual Editor Project
  - XSWT project
  - Essential Data
  - Essential Budget
    - (SWT-based personal finance solution)

- Committer on:
  - Prevayler project
Objectives:

- Survey the Eclipse RCP ecosystem
- Describe other major Eclipse/RCP solutions
- Provide several concrete examples showing techniques Eclipse RCP applications can use in simple or multitier applications
Survey of the Eclipse RCP Ecosystem
Eclipse RCP

• What is it?
  • Eclipse, the application framework
    • Write any application using Eclipse plug-ins
    • Eclipse, minus the “ide-ness”

• Why?
  • Plug-in architecture makes application easy to evolve
  • Server-centric updates, just like web applications
  • A rich client interface provides a higher quality end-user experience
Eclipse RCP

• Tutorials:
  • http://www.eclipse.org/articles/Article-RCP-1/tutorial1.html
  • http://www.eclipse.org/articles/Article-RCP-2/tutorial2.html
Eclipse RCP

- RCP Application Structure

In Eclipse, this is the IDE plug-in. In RCP, you replace this with your own.

(defines your application's default perspective, etc.)
Eclipse RCP

- Dave's quick-and-dirty RCP Example...
  (or how to get started doing RCP without reading the tutorials first)

Plug-in framework

- SWT Plug-in
- JFace/Workbench Plug-in
- RCP "Main Application" Plug-in
- Other RCP Application Plug-ins

Leave this as the Eclipse IDE Plug-in

Only include the necessary Eclipse plug-ins to satisfy dependencies
The main solutions in the Eclipse RCP ecosystem

- The Framework:
  - Eclipse RCP itself
- Front-end tools:
  - Visual Editor Project
  - XSWT
  - SWT Designer
- Data-binding frameworks:
  - Eclipse's JFace framework
  - Essential Data
- Lesser-known back end tools:
  - db4objects
  - Prevayler
Visual Editor Project
(http://www.eclipse.org/vep)

- A GUI builder for Java and Swing
- Edits Java source code
  - “Code-assist on steroids”
Visual Editor Project
(http://www.eclipse.org/vep)

- How do I use it for Rich Client Platform applications?

- Example...
  - (Always edit a Composite)
The main solutions in the Eclipse RCP ecosystem

• The Framework:
  • Eclipse RCP itself
• Front-end tools:
  • Visual Editor Project
  • XSWT
  • SWT Designer
• Data-binding frameworks:
  • Eclipse's JFace framework
  • Essential Data
• Lesser-known back end tools:
  • db4objects
  • Prevayler
XSWT
(http://xswt.sf.net)

- What is it?
  - XML language for SWT page layouts

- Why XSWT?
  - 33% reduction in code size
  - More readable and maintainable than Java
  - 1:1 mapping between Java code and XSWT elements: easy to learn
  - Either embed XSWT engine or compile XSWT to Java
XSWT
(http://xswt.sf.net)

<composite>
  <layoutData x:class="gridData"
    grabExcessHorizontalSpace="true"
    grabExcessVerticalSpace="true"
    horizontalAlignment="GridData.FILL"
    verticalAlignment="GridData.FILL"/>

  <layout x:class="gridLayout" numColumns="2"
    makeColumnsEqualWidth="true"/>

  <x:children>
    <!-- The Path label -->
    <label x:id="Path" text="/"
      <layoutData x:class="gridData"
        grabExcessHorizontalSpace="true"
        horizontalAlignment="GridData.FILL"
        verticalAlignment="GridData.FILL"/>

    </label>
  </x:children>
</composite>
The main solutions in the Eclipse RCP ecosystem

- The Framework:
  - Eclipse RCP itself
- Front-end tools:
  - Visual Editor Project
  - XSWT
  - SWT Designer
- Data-binding frameworks:
  - Eclipse's JFace framework
  - Essential Data
- Lesser-known back end tools:
  - db4objects
  - Prevayler
SWT Designer
(http://www.swt-designer.com/)

- Why SWT Designer?
  - Very high-quality commercial GUI builder for SWT
  - Edits Java source code
The main solutions in the Eclipse RCP ecosystem

- The Framework:
  - Eclipse RCP itself
- Front-end tools:
  - Visual Editor Project
  - XSWT
  - SWT Designer
- Data-binding frameworks:
  - Eclipse's JFace framework
  - Essential Data
- Lesser-known back end tools:
  - db4objects
  - Prevayler
Eclipse's JFace framework

**Advantages**
- Included with Eclipse
- A very general design

**Disadvantages**
- Because the design is so general, you sometimes have to write a lot of code
- Uses SWT's Table widget, which is difficult to use to provide a high-quality cross-platform end-user experience
Eclipse's JFace framework

- Basic TableViewer usage:

  ```java
  JFace
  TableViewer
  ```

  ```java
  ContentProvider
  LabelProvider
  ```

  ```java
  setInput(Object)
  ```

  ```java
  inputChanged()
  ```

  ```java
  getElements(Object)
  ```

  ```java
  Object[]
  ```

  ```java
  forEachElement { getColumnText(Object o, int index) }
  ```

  ```java
  String
  ```

  You write these:
The main solutions in the Eclipse RCP ecosystem

- **The Framework:**
  - Eclipse RCP itself
- **Front-end tools:**
  - Visual Editor Project
  - XSWT
  - SWT Designer
- **Data-binding frameworks:**
  - Eclipse's JFace framework
  - **Essential Data**
- **Lesser-known back end tools:**
  - db4objects
  - Prevayler
Essential Data (http://essentialdata.sf.net)

- **What is it?**
  - Data binding **and validation** framework for SWT
  - Works with any data model layer that uses POJOs

- **Why Essential Data?**
  - Much easier than JFace
    - More user-friendly data grid
    - Picture strings
    - Regular expression validators
    - Automatically handles 1:M relationships

- Most of it is licensed under the GPL (as-of January, 2005)
Essential Data (http://essentialdata.sf.net)

- Binds properties of plain old Java objects (POJOS) to regular SWT controls.

- Uses the Model-View-Controller (MVC) pattern:
  - Fundamentally, Essential Data is a library of **generic MVC controller** objects

The MVC pattern as implemented by Essential Data:

```
Text  Events  POJOFieldAdapter  get/set  Object property
```

<table>
<thead>
<tr>
<th>SWT Text object (View)</th>
<th>Essential Data controller object</th>
<th>Property of a POJO (Model)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Essential Data
(http://essentialdata.sf.net)

- Binds properties of plain old Java objects (POJOS) to regular SWT controls using the MVC pattern
  Minimally, only a getter/setter are required for property read and write.
  (Only a getter is required for read-only fields.)

  ```java
  public class Friend {
    private String name;

    public String getName() {
      return name;
    }

    public void setName(String name) {
      this.name = name;
    }
  }
  ```
Essential Data (http://essentialdata.sf.net)

- Binds properties of plain old Java objects (POJOS) to regular SWT controls using the MVC pattern

```java
public class FriendEditor extends SWTApplication {
    protected void setupUI(Shell parent) {
        // Model (a plain old Java Object)...
        Friend person = new Friend();
        // View (just standard SWT controls)...
        parent.setLayout(new GridLayout(2, false));
        new Label(parent, SWT.NULL).setText("First Name:");
        Text name = new Text(parent, SWT.BORDER);
        name.setLayoutData(new GridData(GridData.FILL_HORIZONTAL | GridData.GRAB_HORIZONTAL));
        // Controller...
        new POJOFieldAdapter(name, "Name", person);
    }

    public static void main(String[] args) {
        new View();
    }
}
```
Essential Data (http://essentialdata.sf.net)

- Binds properties of plain old Java objects (POJOS) to regular SWT controls using the MVC pattern

Everything else builds on this simple platform

✦ Editing whole objects with many properties
✦ Editing collections of objects using a table or slider
✦ Field and object-level validation
✦ Picture strings, regular expression validation
✦ Master-detail relationships
✦ Much more...
The main solutions in the Eclipse RCP ecosystem

- **The Framework:**
  - Eclipse RCP itself
- **Front-end tools:**
  - Visual Editor Project
  - XSWT
  - SWT Designer
- **Data-binding frameworks:**
  - Eclipse's JFace framework
  - Essential Data
- **Lesser-known back end tools:**
  - db4objects
  - Prevayler
db4objects
(www.db4o.com)

- The open-source object database
  - Choice of GPL or very inexpensive commercial license
  - Native Java

- Why db4objects?
  - When you need to persist data on the client (especially eRCP)
  - Very performant
  - Easy to store complex object structures
  - Fully transactional (ACID)
  - Supports object schema evolution
  - Includes object-oriented query languages (QBE and SODA)
  - Works with plain old Java objects (POJOS)
  - Small footprint (~300 kilobyte JAR)
db4objects
(www.db4o.com)

- To store a Pilot object:

```java
ObjectContainer db=Db4o.openFile(Util.YAPFILENAME);
try {
    Pilot pilot1=new Pilot("Michael Schumacher",100);
    db.set(pilot1);
} finally {
    db.close();
}
```
To retrieve all pilots:

```java
ObjectContainer db=Db4o.openFile(Util.YAPFILENAME);
try {
    ObjectSet result=db.get(Pilot.class);
    while (result.hasNext())
        System.out.println(result.next());
}
finally {
    db.close();
}
```
To retrieve a pilot by name using query by example:

```java
ObjectContainer db = Db4o.openFile(Util.YAPFILENAME);
try {
    Pilot example = new Pilot("Michael Schumacher", 0);
    ObjectSet result = db.get(example);
    if (result.hasNext())
        Pilot firstMatch = (Pilot) result.next();
} finally {
    db.close();
}
```
To update a pilot:

```java
ObjectContainer db=Db4o.openFile(Util.YAPFILENAME);
try {
    ObjectSet result=db.get(new Pilot("Michael Schumacher",0));
    Pilot found=(Pilot)result.next();

    found.addPoints(11);
    db.set(found);
}
finally {
    db.close();
}
```
The main solutions in the Eclipse RCP ecosystem

- **The Framework:**
  - Eclipse RCP itself
- **Front-end tools:**
  - Visual Editor Project
  - XSWT
  - SWT Designer
- **Data-binding frameworks:**
  - Eclipse's JFace framework
  - Essential Data
- **Lesser-known back end tools:**
  - db4objects
  - Prevayler
Prevayler
(http://www.prevayler.org)

- What is it?
  First implementation of the object prevalence design pattern

Object Prevalence:
  Simple object persistence for RAM-based objects
  (1 GB RAM ~ $100US)
  Insanely fast performance

Worth considering:
  For small to medium-sized data sets
  For either web service or isolated applications
Prevayler
(http://www.prevayler.org)

• The Object Prevalence design pattern:
  • Every OO system has a start state.
    \[ S_{(0)} \]
  • Method calls transform that start state into subsequent states
    \[ f_{(0)}(S_{(0)}, \text{arguments}_{(0)}) = S_{(1)} \]
    \[ f_{(1)}(S_{(1)}, \text{arguments}_{(1)}) = S_{(2)} \]
    ...
    \[ f_{(n-1)}(S_{(n-1)}, \text{arguments}_{(n-1)}) = S_{(n)} \]
The Object Prevalence design pattern:
- Every OO system has a start state.
  \[ S_{(0)} \]
- Method calls transform that start state into subsequent states
  \[ f_{(0)}(S_{(0)}, \text{arguments}_{(0)}) = S_{(1)} \]
  \[ f_{(1)}(S_{(1)}, \text{arguments}_{(1)}) = S_{(2)} \]
  ...  
  \[ f_{(n-1)}(S_{(n-1)}, \text{arguments}_{(n-1)}) = S_{(n)} \]
- In order to return to an arbitrary state \( S_{(n)} \) from any prior state \( S_{(n-m)} \), all you have to do is to apply all the same method calls starting with \( S_{(n-m)} \) that were applied the first time, in the same order, with the same arguments.
  
  (“Arguments” here includes any global state utilized by \( f \); otherwise we would break the mathematical definition of a function.)
Prevayler
(http://www.prevayler.org)

• Implementing the Object Prevalence design pattern:
  • Prevayler:
    • Your object graph and Command objects must be Serializable
    • Take snapshots at appropriate times
    • Journal Command objects before applying them

• Many systems already use Serializable Command pattern objects
  • Servlets
  • RMI, SOAP servers

• Around Prevayler, we say that these systems have a “natural transaction barrier”
  • Persistence for free == adding Prevayler into the pipeline these Command objects already travel.
The main solutions in the Eclipse RCP ecosystem (recap)

- The Framework:
  - Eclipse RCP itself
- Front-end tools:
  - Visual Editor Project
  - XSWT
  - SWT Designer
- Data-binding frameworks:
  - Eclipse's JFace framework
  - Essential Data
- Lesser-known back end tools:
  - db4objects
  - Prevayler
Other major Eclipse/RCP solutions

- The SWT library itself
  - [http://www.eclipse.org/swt](http://www.eclipse.org/swt)
- RCPLite
  - Part of Essential Data
  - An Eclipse-like UI with a fraction of the effort
  - No plug-ins or update manager
- RSWT
  - Make SWT work like a server-based remote control application
  - [http://rswt.sf.net](http://rswt.sf.net)
- WebRCP
  - Java Web-start the Eclipse RCP
  - [http://webrcp.sf.net](http://webrcp.sf.net)
- Hibernate, Spring, etc.
  - Other server-side Java frameworks applicable to RCP
Survey of the Eclipse RCP Ecosystem (reprise)

Eclipse Technologies

Eclipse RCP Technologies

- Plug-in Framework
  - SWT
  - JFace
  - Workbench

Visual Editor
- Project

EMF
- Resource framework

Update Manager

Help

- db4objects
  - Prevayler
  - Spring

- Essential Data

- SWT Designer
- XSWT
- RCPLite
- R SWT
- WebRCP
- Hibernate

- Eclipse Technologies
- Eclipse RCP Technologies
  - Plug-in Framework
    - SWT
    - JFace
    - Workbench

- Visual Editor
  - Project

- EMF
  - Resource framework

- Update Manager

- Help
  - db4objects
    - Prevayler
    - Spring

- Essential Data

- SWT Designer
- XSWT
- RCPLite
- R SWT
- WebRCP
- Hibernate

39
Conclusion:

- Benefits of the Eclipse RCP solution
  - Server-based infrastructure
  - Rich-client user experience

- The community is beginning to fill in the gaps required to deliver a complete Eclipse RCP solution.

- We have seen how these solutions can be combined to make interesting Eclipse RCP solutions.