```
package protodebugger.model;

import java.awt.Color;
import java.beans.PropertyChangeListener;
import java.beans.PropertyChangeSupport;
import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import org.eclipse.swt.graphics.RGB;
import org.eclipse.ui.PlatformUI;
import org.eclipse.ui.console.ConsolePlugin;
import org.eclipse.ui.console.IConsole;
import org.eclipse.ui.console.MessageConsole;
import org.eclipse.ui.console.MessageConsoleStream;

import protodebugger.model.descriptors.BooleanFieldDescriptorContainer;
import protodebugger.model.descriptors.EnumFieldDescriptorContainer;
import protodebugger.model.descriptors.FieldDescriptorContainer;
import protodebugger.model.descriptors.MessageFieldDescriptorContainer;
import protodebugger.model.descriptors.NumberFieldDescriptorContainer;
import protodebugger.model.descriptors.TextFieldDescriptorContainer;

import com.google.protobuf.Descriptors;
import com.google.protobuf.GeneratedMessage;
import com.google.protobuf.GeneratedMessage.Builder;
import com.google.protobuf.Message;

public enum ParseProtoMessage {

        INSTANCE;
        private Map<GeneratedMessage, ProtoMessage> members = new
HashMap<GeneratedMessage, ProtoMessage>();
        private PropertyChangeSupport pcs = new PropertyChangeSupport(this);
        private MessageConsole protoConsole;
        private GeneratedMessage current;

        private ParseProtoMessage()
        {
                ConsolePlugin console = ConsolePlugin.getDefault();
                protoConsole = new MessageConsole("PROTO_VIEW", null);
                console.getConsoleManager().addConsoles(new IConsole[]{protoConsole});
        }

        private void printInformation(String info)
        {
                MessageConsoleStream stream = protoConsole.newMessageStream();
                stream.setColor(new org.eclipse.swt.graphics.Color(PlatformUI.getWorkbench
().getActiveWorkbenchWindow().getWorkbench().getDisplay(), new RGB(Color.blue.getRed(),
Color.blue.getGreen(),
                                Color.blue.getBlue())));
                stream.println(info);
                try{
```

```
                        stream.close();
                }catch(IOException ioe)
                {
                        System.err.println("noooooo");
                }
        }

        private void printError(String error)
        {
                MessageConsoleStream stream = protoConsole.newMessageStream();
                stream.setColor(new org.eclipse.swt.graphics.Color(PlatformUI.getWorkbench
().getActiveWorkbenchWindow().getWorkbench().getDisplay(), new RGB(Color.red.getRed(),
Color.red.getGreen(),
                          Color.red.getBlue())));
                stream.println(error);
                try{
                        stream.close();
                }catch(IOException ioe)
                {
                        System.err.println("noooooo");
                }
        }

        public void printMessageToConsole(String msg, boolean error)
        {
                if(error)
                        printError(msg);
                else
                        printInformation(msg);
        }


        public void parse(GeneratedMessage msg)
        {
                printInformation("\t Loading message");
                ArrayList<FieldDescriptorContainer> fields = new
ArrayList<FieldDescriptorContainer>();
                ArrayList<FieldDescriptorContainer> repeated = new
ArrayList<FieldDescriptorContainer>();
                FieldDescriptorContainer container;
                for (Descriptors.FieldDescriptor field : msg.getDescriptorForType()
                                .getFields()) {
                        container = parseFieldDescriptor(field);
                        if(container != null)
                        {
                                fields.add(container);
                                if(field.isRepeated())
                                {
                                        repeated.add(container);
                                }
                        }
                }
                members.put(msg,  new ProtoMessage(msg, fields, repeated));
        }

        public void addChangeListener(PropertyChangeListener pcl)
```

```
        {
                pcs.addPropertyChangeListener(pcl);
        }

        public FieldDescriptorContainer parseFieldDescriptor(Descriptors.FieldDescriptor
field)
        {
                switch (field.getJavaType()) {
                case FLOAT:
                case INT:
                case DOUBLE:
                case LONG:
                        return (new NumberFieldDescriptorContainer(field));
                case STRING:
                case BYTE_STRING:
                        return (new TextFieldDescriptorContainer(field));
                case BOOLEAN:
                        return (new BooleanFieldDescriptorContainer(field));
                case ENUM:
                        return (new EnumFieldDescriptorContainer(field));
                case MESSAGE:
                        return (new MessageFieldDescriptorContainer(field));
                }
                return null;
        }

        public void selectionChange(GeneratedMessage msg)
        {
                current = msg;
                pcs.firePropertyChange("PROTO_CHANGE", null, msg);
        }

        public void sendProtoToConsole()
        {
                printInformation("Sending Proto - " + current.getDescriptorForType
().getName());

                Builder<?> type = (Builder<?>) current.newBuilderForType();

                for ( FieldDescriptorContainer field : getListforMsg(current))
                {
                        field.buildMsg(type);
                }
                Message genMsg = type.build();
                printInformation(genMsg.toString());
                printInformation("Message Sent\n");
        }

        public void removeAddedRepeatedField(FieldDescriptorContainer field)
        {
                removeAddedRepeatedField(field, current);
        }

        public void removeAddedRepeatedField(FieldDescriptorContainer field,
GeneratedMessage msg)
```

builder must just return a message

```
	{
		members.get(msg).removeAddedField(field);
		pcs.firePropertyChange("REMOVE_FIELD", field, msg);
	}

	public void addRepeatedField(FieldDescriptorContainer field)
	{
		addRepeatedField(field, current);
	}

	public void addRepeatedField(FieldDescriptorContainer field, GeneratedMessage msg)
	{
		FieldDescriptorContainer added = parseFieldDescriptor(field.field);
		if(!field.isSubField())
		{
			int index = members.get(msg).getContents().indexOf(field);
			members.get(msg).getContents().add(index+1, added);
		}
		else{
			FieldDescriptorContainer parentField = field.getFieldParent();
			if(parentField != null && parentField instanceof
MessageFieldDescriptorContainer)
			{
				int index = ((MessageFieldDescriptorContainer)
parentField).getMembers().indexOf(field);
				((MessageFieldDescriptorContainer)parentField).addMember
(added, index+1);
			}
		}
		members.get(msg).addAddField(field);
		pcs.firePropertyChange("REPEATED_FIELD", field, added);
	}

	public List<FieldDescriptorContainer> getListforMsg(GeneratedMessage msg){
		printInformation("Retrieving contents for '"+msg.getDescriptorForType
().getName()+"'");
		if(!members.containsKey(msg))
			parse(msg);
		return members.get(msg).getContents();
	}

	public List<FieldDescriptorContainer> getRepeatedforMsg(){
		if(!members.containsKey(current))
			return new ArrayList<FieldDescriptorContainer>();
		return members.get(current).getRepeatedFields();
	}
	public List<FieldDescriptorContainer> getAddedforMsg(){
		if(!members.containsKey(current))
			return new ArrayList<FieldDescriptorContainer>();
		return members.get(current).getAddFields();
	}

	public void printContainments()
	{
		for( GeneratedMessage msg : members.keySet())
		{
```

```
                        System.out.println("NEW Message "+ msg.toString());
                        for(FieldDescriptorContainer field : members.get(msg).getContents
())
                        {
                                System.out.println(field.toString());
                        }
                }
        }

}
```

Just an header file

```
package protodebugger.model;

import java.util.ArrayList;
import java.util.List;

import protodebugger.model.descriptors.FieldDescriptorContainer;

import com.google.protobuf.GeneratedMessage;

public class ProtoMessage {

	private GeneratedMessage genMsg;
	private List<FieldDescriptorContainer> contents;
	private List<FieldDescriptorContainer> repeatedFields;
	private List<FieldDescriptorContainer> addFields;


	public ProtoMessage(GeneratedMessage genMsg,
			List<FieldDescriptorContainer> contents,
			List<FieldDescriptorContainer> repeatedFields) {
		super();
		this.genMsg = genMsg;
		this.contents = contents;
		this.repeatedFields = repeatedFields;
	}
	public GeneratedMessage getGenMsg() {
		return genMsg;
	}
	public void setGenMsg(GeneratedMessage genMsg) {
		this.genMsg = genMsg;
	}
	public List<FieldDescriptorContainer> getContents() {
		return contents;
	}
	public void setContents(List<FieldDescriptorContainer> contents) {
		this.contents = contents;
	}
	public List<FieldDescriptorContainer> getRepeatedFields() {
		return repeatedFields;
	}
	public void setRepeatedFields(List<FieldDescriptorContainer> repeatedFields) {
		this.repeatedFields = repeatedFields;
	}
	public void addAddField(FieldDescriptorContainer add)
	{
		if(addFields == null)
			addFields = new ArrayList<FieldDescriptorContainer>();
		addFields.add(add);
	}
	public void removeAddedField(FieldDescriptorContainer add)
	{
		if(addFields == null)
			return;
		addFields.remove(add);
	}
	public List<FieldDescriptorContainer> getAddFields()
```

```
        {
                return addFields;
        }


}
```

```
package protodebugger.model.descriptors;

import org.eclipse.swt.SWT;
import org.eclipse.swt.widgets.Composite;
import org.eclipse.swt.widgets.Text;
import org.eclipse.swt.widgets.Widget;

import com.google.protobuf.Descriptors;
import com.google.protobuf.GeneratedMessage.Builder;

public class NumberFieldDescriptorContainer extends FieldDescriptorContainer {
	private Text textField;
	public NumberFieldDescriptorContainer(Descriptors.FieldDescriptor field)
	{
		super(field);
	}

	@Override
	public void setValue(Object value)
	{
		this.value = value;
	}

	@Override
	public Object getValue()
	{
		if(value != null)
			return value.toString();
		else if(defaultValue != null)
			return defaultValue.toString();
		else
			return "";
	}


	@Override
	public String toString()
	{
		return "NumberField name = " + name;
	}

	@Override
	public Widget getWidget(Composite parent)
	{
		if(textField == null)
		{
			textField = new Text(parent, SWT.BORDER);
			textField.setText((String)getValue());
		} else if(textField.getParent() != parent)
		{
			textField.setParent(parent);
		}
		return textField;
	}
```

```
	@Override
	public boolean buildMsg(Builder<?> build){
		if(field.isOptional() && textField.getText().equals(""))
			return false;

		switch(field.getJavaType())
		{
		case INT:
			if(field.isRepeated())
				build.addRepeatedField(field, Integer.parseInt
(textField.getText()));
			else
				build.setField(field, Integer.parseInt(textField.getText
()));
			break;
		case FLOAT:
			if(field.isRepeated())
				build.addRepeatedField(field, Float.parseFloat
(textField.getText()));
			else
				build.setField(field, Float.parseFloat(textField.getText
()));
			break;
		case DOUBLE:
			if(field.isRepeated())
				build.addRepeatedField(field, Double.parseDouble
(textField.getText()));
			else
				build.setField(field, Double.parseDouble(textField.getText
()));
			break;
		case LONG:
			if(field.isRepeated())
				build.addRepeatedField(field, Long.parseLong
(textField.getText()));
			else
				build.setField(field, Long.parseLong(textField.getText()));
			break;
		}
		return true;
	}

	@Override
	public Composite getParent() {
		if(textField != null)
			return textField.getParent();
		return null;
	}

}
```

```
package protodebugger.model.descriptors;


import org.eclipse.swt.SWT;
import org.eclipse.swt.widgets.Button;
import org.eclipse.swt.widgets.Composite;
import org.eclipse.swt.widgets.Widget;

import com.google.protobuf.Descriptors;
import com.google.protobuf.GeneratedMessage.Builder;

public class BooleanFieldDescriptorContainer extends FieldDescriptorContainer {

	private Button check;

	public BooleanFieldDescriptorContainer(Descriptors.FieldDescriptor field)
	{
		super(field);
	}

	@Override
	public void setValue(Object value)
	{
		this.value = value;
	}

	@Override
	public Object getValue()
	{
		if(value != null)
			return value.toString();
		else if(defaultValue != null)
			return defaultValue.toString();
		else
			return "";
	}

	@Override
	public String toString()
	{
		return "BooleanField name = " + name;
	}

	@Override
	public Widget getWidget(Composite parent)
	{
		if(check == null)
			check = new Button(parent, SWT.CHECK);
		else if(check.getParent() != parent)
		{
			check.setParent(parent);
		}
		return check;
	}
	@Override
	public boolean buildMsg(Builder<?> build){
```

```
                if(field.isRepeated())
                        build.addRepeatedField(field, check.getSelection());
                else
                        build.setField(field, check.getSelection());
                return true;
        }

        @Override
        public Composite getParent() {
                if(check != null)
                        return check.getParent();
                return null;
        }

}
```

```
package protodebugger.model.descriptors;

import org.eclipse.swt.SWT;
import org.eclipse.swt.widgets.Combo;
import org.eclipse.swt.widgets.Composite;
import org.eclipse.swt.widgets.Widget;

import com.google.protobuf.Descriptors;
import com.google.protobuf.Descriptors.EnumValueDescriptor;
import com.google.protobuf.GeneratedMessage.Builder;

public class EnumFieldDescriptorContainer extends FieldDescriptorContainer {
        private Combo comboField;
        public EnumFieldDescriptorContainer(Descriptors.FieldDescriptor field)
        {
                super(field);
        }
        @Override
        public Object getValue()
        {
                if(value != null)
                        return value.toString();
                else if(defaultValue != null)
                        return ((EnumValueDescriptor)defaultValue).getName();
                else
                        return field.getEnumType().getValues().get(0).toString();
        }

        @Override
        public void setValue(Object value)
        {
                if(value instanceof String)
                {
                        for (EnumValueDescriptor evd : field.getEnumType().getValues()) {
                                if(evd.getName().equals(value))
                                {
                                        value = evd;
                                        return;
                                }
                        }
                }
        }

        @Override
        public String toString()
        {
                return "EnumField name = "+name;
        }

        @Override
        public Widget getWidget(Composite parent)
        {
                if(comboField == null)
                {
                        comboField = new Combo(parent, SWT.READ_ONLY);
                        comboField.setItems(getValues());
```

```
                        if(defaultValue != null)
                                comboField.select(((EnumValueDescriptor)
defaultValue).getIndex());
                } else if(comboField.getParent() != parent)
                {
                        comboField.setParent(parent);
                }

                return comboField;
        }

        public String[] getValues()
        {
                int array_size = field.getEnumType().getValues().size();
                if(field.isOptional())
                        array_size += 1;
                String[] ret = new String[array_size];
                for (EnumValueDescriptor evd : field.getEnumType().getValues()) {
                        ret[evd.getIndex()] = evd.getName();
                }
                if(field.isOptional())
                        ret[array_size-1] = "";
                return ret;
        }

        @Override
        public boolean buildMsg(Builder<?> build){
                if(field.isOptional() && comboField.getText().equals(""))
                        return false;
                if(field.isRepeated())
                        build.addRepeatedField(field, field.getEnumType().findValueByName
(comboField.getText()));
                else
                        build.setField(field,field.getEnumType().findValueByName
(comboField.getText()));
                return true;
        }
        @Override
        public Composite getParent() {
                if(comboField != null)
                        return comboField.getParent();
                return null;
        }

}
```

```
package protodebugger.model.descriptors;

import org.eclipse.swt.widgets.Composite;
import org.eclipse.swt.widgets.Widget;

import com.google.protobuf.Descriptors;
import com.google.protobuf.GeneratedMessage.Builder;

public abstract class FieldDescriptorContainer {

	public String name;
	protected Object defaultValue, value;
	public  Descriptors.FieldDescriptor field;
	protected boolean subField = false;
	protected FieldDescriptorContainer parent;
	public FieldDescriptorContainer(Descriptors.FieldDescriptor field)
	{
		this.field = field;
		this.name = field.getName();
		if(field.hasDefaultValue())
			defaultValue = field.getDefaultValue();
	}


	public abstract Object getValue();
	public abstract boolean buildMsg(Builder<?> build);
	public abstract void setValue(Object value);
	@Override
	public abstract String toString();
	public abstract Widget getWidget(Composite parent);
	public abstract Composite getParent();
	public boolean isSubField(){return subField;}
	public void setSubField(boolean sub){subField = sub;}
	public void setFieldParent(FieldDescriptorContainer parent){this.parent = parent;}
	public FieldDescriptorContainer getFieldParent(){if(subField)return parent;return
null;}
}
```

```
package protodebugger.model.descriptors;

import java.util.ArrayList;
import java.util.List;

import org.eclipse.swt.widgets.Composite;
import org.eclipse.swt.widgets.Widget;

import protodebugger.model.ParseProtoMessage;

import com.google.protobuf.Descriptors;
import com.google.protobuf.GeneratedMessage.Builder;

public class MessageFieldDescriptorContainer extends FieldDescriptorContainer {
        private ArrayList<FieldDescriptorContainer> subMembers;
        private Composite parent;
        public MessageFieldDescriptorContainer(Descriptors.FieldDescriptor field)
        {
                super(field);
                subMembers = new ArrayList<FieldDescriptorContainer>();
                parseMessage();

        }

        private void parseMessage()
        {
                for (Descriptors.FieldDescriptor inner : field.getMessageType()
                                .getFields()) {
                        FieldDescriptorContainer subField =
ParseProtoMessage.INSTANCE.parseFieldDescriptor(inner);
                        subField.setSubField(true);
                        subField.setFieldParent(this);
                        subMembers.add(subField);
                }
        }

        public void addMember(FieldDescriptorContainer field, int index)
        {
                subMembers.add(index, field);
        }

        public List<FieldDescriptorContainer> getMembers()
        {
                return subMembers;
        }

        @Override
        public void setValue(Object value)
        {

        }

        @Override
        public Widget getWidget(Composite parent)
        {
                return null;
```

```
	}

	@Override
	public Object getValue()
	{
		return null;
	}

	@Override
	public String toString()
	{
		StringBuilder build = new StringBuilder("MessageField name = " + name +
"\n");
		for(FieldDescriptorContainer field : subMembers)
		{
			build.append("\t"+field.toString() + "\n");
		}
		return build.toString();
	}
	@Override
	public boolean buildMsg(Builder<?> build){
		boolean buildIt = false;
		Builder<?> innerBuild = (Builder<?>)build.newBuilderForField(field);
		for(FieldDescriptorContainer inner : getMembers())
		{
			System.out.println("Building " + inner.toString());
			inner.buildMsg(innerBuild);
		}
		if(field.isRepeated())
			build.addRepeatedField(field, innerBuild.build());
		else
			build.setField(field, innerBuild.build());

	return true;
	}

	public void setParent(Composite parent)
	{
		this.parent = parent;
	}
	@Override
	public Composite getParent() {
		// TODO Auto-generated method stub
		return parent;
	}


}
```

```
package protodebugger.views;

import java.beans.PropertyChangeEvent;
import java.beans.PropertyChangeListener;
import java.util.List;

import org.eclipse.swt.SWT;
import org.eclipse.swt.events.ExpandEvent;
import org.eclipse.swt.events.ExpandListener;
import org.eclipse.swt.widgets.Composite;
import org.eclipse.swt.widgets.Control;
import org.eclipse.swt.widgets.ExpandBar;
import org.eclipse.swt.widgets.ExpandItem;
import org.eclipse.ui.PlatformUI;
import org.eclipse.ui.part.ViewPart;

import protodebugger.Activator;
import protodebugger.model.ParseProtoMessage;
import protodebugger.model.descriptors.FieldDescriptorContainer;
import protodebugger.model.descriptors.MessageFieldDescriptorContainer;

import com.google.protobuf.GeneratedMessage;

/**
 * This sample class demonstrates how to plug-in a new workbench view. The view
 * shows data obtained from the model. The sample creates a dummy model on the
 * fly, but a real implementation would connect to the model available either in
 * this or another plug-in (e.g. the workspace). The view is connected to the
 * model using a content provider.
 * <p>
 * The view uses a label provider to define how model objects should be
 * presented in the view. Each view can present the same model objects using
 * different labels and icons, if needed. Alternatively, a single label provider
 * can be shared between views in order to ensure that objects of the same type
 * are presented in the same way everywhere.
 * <p>
 */

public class ProtoViewer extends ViewPart implements PropertyChangeListener{

	/**
	 * The ID of the view as specified by the extension.
	 */
	public static final String ID = "protodebugger.views.ProtoViewer";
	private ExpandBar expandBar;
	private ExpandBar currentBar;
	private int SPACING = 5;

	/**
	 * The constructor.
	 */
	public ProtoViewer() {
		ParseProtoMessage.INSTANCE.addChangeListener(this);
	}
```

```
	private void addSubField(FieldDescriptorContainer field)
	{

		ExpandItem item = new ExpandItem(currentBar, SWT.NONE);
		item.setText(field.name.replace("_", " "));
		item.setData(field);
		item.setExpanded(false);
		item.setControl((Control)field.getWidget(currentBar));
		item.setHeight(item.getControl().computeSize(SWT.DEFAULT, SWT.DEFAULT).y);
		//item.setImage(Activator.getImageDescriptor("icons/
sample.gif").createImage());
	}

	private void addMessageField(MessageFieldDescriptorContainer field) {
		final ExpandItem item = new ExpandItem(currentBar, SWT.NONE);
		item.setText(field.name.replace("_", " "));
		item.setExpanded(false);
		item.setData(field);
		final ExpandBar innerBar = new ExpandBar(currentBar, SWT.NONE);
		innerBar.setSpacing(SPACING);
		item.setControl(innerBar);
		field.setParent(currentBar);
		currentBar = innerBar;
		for (FieldDescriptorContainer innerField : field.getMembers()) {

			if (innerField instanceof MessageFieldDescriptorContainer) {
				addMessageField((MessageFieldDescriptorContainer)
innerField);
				currentBar = innerBar;
			} else
				addSubField(innerField);
		}
		item.setHeight(item.getControl().computeSize(SWT.DEFAULT, SWT.DEFAULT).y);
		innerBar.addExpandListener(new ExpandListener() {

			@Override
			public void itemExpanded(ExpandEvent e) {
				updateBar();
			}

			@Override
			public void itemCollapsed(ExpandEvent e) {
				updateBar();
			}

			private void updateBar() {
				PlatformUI.getWorkbench().getActiveWorkbenchWindow
().getShell()
					.getDisplay().asyncExec(new Runnable() {

						@Override
						public void run() {
							int height = 0;
							for (ExpandItem
nested_item : innerBar
.getItems
```

```
()) {
                                                                    //if
(nested_item.getExpanded())
                                                                    height +=
(nested_item.getControl()
                                                                                .computeSize
(SWT.DEFAULT,
SWT.DEFAULT).y)
                                                                                +
nested_item.getHeight();
                                                                    /*else
                                                                            height -=
(nested_item.getControl()
                                                                                        .com
(SWT.DEFAULT,
SWT.DEFAULT).y)
+ nested_item.getHeight();*/
                                                                }
                                                                item.setHeight(height );
                                                            }
                                                        });

                        }
                });
        }

        /**
         * This is a callback that will allow us to create the viewer and initialize
         * it.
         */
        public void createPartControl(Composite parent) {
        //      GeneratedMessage msg = TacticalMessage.getDefaultInstance();
                expandBar = new ExpandBar(parent, SWT.BORDER | SWT.V_SCROLL);
                expandBar.setSpacing(SPACING);
        }
        public void selectionChanged(GeneratedMessage msg)
        {
                for(ExpandItem item: expandBar.getItems())
                {
                        item.setExpanded(false);
                        item.dispose();
                }
                createExpandItems(msg);
        }

        private void createExpandItems(GeneratedMessage msg) {
                expandBar.setData(msg);
                currentBar = expandBar;
                List<FieldDescriptorContainer> fields = ParseProtoMessage.INSTANCE
                                .getListforMsg(msg);

                for (FieldDescriptorContainer field : fields) {
                        if (field instanceof MessageFieldDescriptorContainer) {
```

```
                                addMessageField((MessageFieldDescriptorContainer) field);
                                currentBar = expandBar;
                        } else
                                addSubField(field);
                }

        }

        /**
         * Passing the focus request to the viewer's control.
         */
        public void setFocus() {

        }

        public void propertyChange(PropertyChangeEvent evt){
                if(evt.getPropertyName().equals("PROTO_CHANGE") || evt.getPropertyName
().equals("REMOVE_FIELD"))
                {
                        GeneratedMessage msg = (GeneratedMessage)evt.getNewValue();
                        selectionChanged(msg);
                }
                else if(evt.getPropertyName().equals("REPEATED_FIELD"))
                {
                        FieldDescriptorContainer field = (FieldDescriptorContainer)
evt.getOldValue();
                        Composite comp = field.getParent();

                        if(comp != null)
                        {
                                currentBar = (ExpandBar)comp;
                                FieldDescriptorContainer added  =
(FieldDescriptorContainer) evt.getNewValue();
                                if(added instanceof MessageFieldDescriptorContainer)
                                        addMessageField((MessageFieldDescriptorContainer)
added);
                                else
                                        addSubField(added);
                        }
                }
        }

}
```

```
package protodebugger.model.descriptors;

import org.eclipse.swt.SWT;
import org.eclipse.swt.widgets.Composite;
import org.eclipse.swt.widgets.Text;
import org.eclipse.swt.widgets.Widget;

import com.google.protobuf.ByteString;
import com.google.protobuf.Descriptors;
import com.google.protobuf.GeneratedMessage.Builder;


public class TextFieldDescriptorContainer extends FieldDescriptorContainer {

	private Text textField;

	public TextFieldDescriptorContainer(Descriptors.FieldDescriptor field)
	{
		super(field);
	}


	@Override
	public void setValue(Object value)
	{
		this.value = value;
	}

	@Override
	public Object getValue()
	{
		if(value != null)
			return value.toString();
		else if(defaultValue != null)
			return defaultValue.toString();
		else
			return "";
	}

	@Override
	public String toString()
	{
		return "TextField name = "+name;
	}
	@Override
	public Widget getWidget(Composite parent)
	{
		if(textField == null)
		{
			textField = new Text(parent, SWT.BORDER);
			textField.setText((String)getValue());
		} else if(textField.getParent() != parent)
		{
			textField.setParent(parent);
		}
```

```
                return textField;
        }

        @Override
        public boolean buildMsg(Builder<?> build){
                if(field.isOptional() && textField.getText().equals(""))
                        return false;
                switch(field.getJavaType())
                {
                case BYTE_STRING:
                        if(field.isRepeated())
                                build.addRepeatedField(field, ByteString.copyFrom
(textField.getText().getBytes()));
                        else
                                build.setField(field, ByteString.copyFrom(textField.getText
().getBytes()));
                        break;
                default:
                        if(field.isRepeated())
                                build.addRepeatedField(field, textField.getText());
                        else
                                build.setField(field, textField.getText());
                }

                return true;
        }


        @Override
        public Composite getParent() {
                if(textField != null)
                        return textField.getParent();
                return null;
        }

}
```