

```
14986 package ui;
14987 import java.awt.GridLayout;
14988 import java.awt.event.ActionEvent;
14989 import java.awt.event.MouseAdapter;
14990 import java.awt.event.MouseEvent;
14991 import java.util.Map;
14992 import javax.swing.AbstractAction;
14993 import javax.swing.BorderFactory;
14994 import javax.swing.ComboBoxModel;
14995 import javax.swing.DefaultComboBoxModel;
14996 import javax.swing.JButton;
14997 import javax.swing.JComboBox;
14998 import javax.swing.JLabel;
14999 import javax.swing.JList;
15000 import javax.swing.JPanel;
15001 import javax.swing.JScrollPane;
15002 import javax.swing.JTable;
15003 import javax.swing.JTextArea;
15004 import javax.swing.ListModel;
15005 import javax.swing.ListSelectionModel;
15006 import javax.swing.SwingUtilities;
15007 import javax.swing.WindowConstants;
15008 import javax.swing.border.BevelBorder;
15009 import javax.swing.border.LineBorder;
15010 import javax.swing.event.ListSelectionEvent;
15011 import javax.swing.event.ListSelectionListener;
15012 import javax.swing.table.DefaultTableModel;
15013 import javax.swing.table.TableModel;
15014 import cn.edu.ustb.webservicetester.model.OperationInfo;
15015 import cn.edu.ustb.webservicetester.model.ParameterInfo;
15016 public class PreferenceFrame extends javax.swing.JFrame {
15017     private JPanel jPanel1;
15018     private JScrollPane jScrollPane1;
15019     private JScrollPane jScrollPane2;
15020     private AbstractAction nextAction;
15021     private JButton jButtonPre;
15022     private JButton jButtonNext;
15023     private JComboBox testStrategyJComboBox;
15024     private JLabel testStrategyJLabel;
15025     private DefaultTableModel paraInfoTableModel;
15026     private JTable paraInfoJTable;
15027     private JTextArea paraResJTextArea;
15028     private JLabel paraResJLabel;
15029     private JLabel paraInfoJLabel;
15030     private AbstractAction operationSelectionAction;
15031     private JComboBox operationComboBox;
15032     private JLabel opSelectionJLabel;
15033     private java.util.List<OperationInfo> opInfoList;
15034     private OperationInfo selectedOperationInfo;
15035     private java.util.List<ParameterInfo> paraInfoList;
```

```
15036 private java.util.Map<String, String> setting = null;
15037 public static void main(String[] args) {
15038     SwingUtilities.invokeLater(new Runnable() {
15039         public void run() {
15040             PreferenceFrame inst = new PreferenceFrame();
15041             inst.setLocationRelativeTo(null);
15042             inst.setVisible(true);
15043         }
15044     });
15045 }
15046 public PreferenceFrame() {
15047     super();
15048     initGUI();
15049 }
15050 public PreferenceFrame(Map<String, String> setting, java.util.List list) {
15051     super();
15052     this.setting = setting;
15053     initGUI(list);
15054 }
15055 private void initGUI() {
15056     initGUI(null);
15057 }
15058 private void initGUI(java.util.List<OperationInfo> oil) {
15059     this.opInfoList = oil;
15060     try {
15061         setTitle("操作选择");
15062         setResizable(false);
15063         GridLayout thisLayout = new GridLayout(1, 1);
15064         getContentPane().setLayout(thisLayout);
15065         setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
15066         {
15067             JPanel1 = new JPanel();
15068             getContentPane().add(JPanel1);
15069             JPanel1.setLayout(null);
15070             JPanel1.setBackground(new java.awt.Color(228,236,243));
15071             JPanel1.setPreferredSize(new java.awt.Dimension(506, 349));
15072             {
15073                 opSelectionJLabel = new JLabel();
15074                 JPanel1.add(opSelectionJLabel);
15075                 opSelectionJLabel.setText("\u8bf7\u9009\u62e9\u5f85\u6d4b\u8bd5\u7684\u64cd\u4f5c");
15076                 opSelectionJLabel.setBounds(12, 36, 128, 17);
15077             }
15078             {
15079                 ComboBoxModel operationSelectionComboBoxModel;
15080                 if (opInfoList == null)
15081                     operationSelectionComboBoxModel =
15082                         new DefaultComboBoxModel(
15083                             new String[] { "Item One", "Item Two" });
15084                 else {
15085                     String[] opNames = new String[opInfoList.size()];
```

```

15086         for (int i = 0; i < opNames.length; ++i) {
15087             opNames[i] = opInfoList.get(i).getOpName();
15088         }
15089         operationSelectionComboBoxModel = new
15090 DefaultComboBoxModel(opNames);
15091     }
15092     operationComboBox = new JComboBox();
15093     jPanel1.add(operationComboBox);
15094     operationComboBox.setModel(operationSelectionComboBoxModel);
15095     operationComboBox.setBounds(140, 32, 162, 24);
15096     operationComboBox.setAction(getOperationSelectionAction());
15097 }
15098 {
15099     jScrollPane1 = new JScrollPane();
15100     jPanel1.add(jScrollPane1);
15101     jScrollPane1.setBounds(24, 101, 183, 132);
15102     jScrollPane1.setViewportViewView(getParaInfoJTable());
15103 }
15104 {
15105     jScrollPane2 = new JScrollPane();
15106     jPanel1.add(jScrollPane2);
15107     jPanel1.add(getJLabelParameterInformation());
15108     jPanel1.add(getParaResJLabel());
15109     jPanel1.add(getTestStrategyJLabel());
15110     jPanel1.add(getTestStrategyJComboBox());
15111     jPanel1.add(getJButtonNext());
15112     jPanel1.add(getJButtonPre());
15113     jScrollPane2.setBounds(290, 101, 183, 132);
15114     jScrollPane2.setViewportViewView(getParaResJTextArea());
15115 }
15116 }
15117 pack();
15118 this.setSize(500, 444);
15119 } catch (Exception e) {
15120     e.printStackTrace();
15121 }
15122 }
15123 private AbstractAction getOperationSelectionAction() {
15124     if(operationSelectionAction == null) {
15125         operationSelectionAction = new AbstractAction("", null) {
15126             public void actionPerformed(ActionEvent evt) {
15127                 int opIndexSelected = operationComboBox.getSelectedIndex();
15128                 selectedOperationInfo = opInfoList.get(opIndexSelected);
15129                 paraInfoList = selectedOperationInfo.getParaList();
15130                 int oldPInfoCount = paraInfoTableModel.getRowCount();
15131                 while (oldPInfoCount-- > 0) {
15132                     paraInfoTableModel.removeRow(0);
15133                 }
15134                 for (ParameterInfo pi : paraInfoList) {
15135                     paraInfoTableModel.addRow(new String[] {pi.getName(),

```

```
15136 pi.getClassInfo()));
15137     }
15138     }
15139     };
15140     }
15141     return operationSelectionAction;
15142 }
15143 private JLabel getJLabelParameterInformation() {
15144     if(paraInfoJLabel == null) {
15145         paraInfoJLabel = new JLabel();
15146         paraInfoJLabel.setText("\u8f93\u5165\u53c2\u6570");
15147         paraInfoJLabel.setBounds(24, 73, 61, 17);
15148     }
15149     return paraInfoJLabel;
15150 }
15151 private JLabel getParaResJLabel() {
15152     if(paraResJLabel == null) {
15153         paraResJLabel = new JLabel();
15154         paraResJLabel.setText("\u53c2\u6570\u8bf4\u660e");
15155         paraResJLabel.setBounds(274, 73, 64, 17);
15156     }
15157     return paraResJLabel;
15158 }
15159 private JTextArea getParaResJTextArea() {
15160     if(paraResJTextArea == null) {
15161         paraResJTextArea = new JTextArea();
15162         paraResJTextArea.setEditable(false);
15163     }
15164     return paraResJTextArea;
15165 }
15166 private JTable getParaInfoJTable() {
15167     if (paraInfoJTable == null) {
15168         String[] name = new String[] {"参数名", "数据类型"};
15169         int opIndexSelected = operationComboBox.getSelectedIndex();
15170         OperationInfo opInfo = opInfoList.get(opIndexSelected);
15171         paraInfoList = opInfo.getParaList();
15172         paraInfoTableModel = new DefaultTableModel(null, name);
15173         for (ParameterInfo pi : paraInfoList) {
15174             paraInfoTableModel.addRow(new String[] {pi.getName(), pi.getClassInfo()});
15175         }
15176         paraInfoJTable = new JTable(paraInfoTableModel) {
15177             public boolean isCellEditable(int row, int column) {
15178                 return false;
15179             }
15180         };
15181         paraInfoJTable.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
15182         paraInfoJTable.addMouseListener(new MouseAdapter() {
15183             public void mouseClicked(MouseEvent evt) {
15184                 int selectedRow = paraInfoJTable.getSelectedRow();
15185                 if (paraResJTextArea != null && paraInfoList != null &&
```

```
15186     selectedRow >= 0)
15187         paraResJTextArea.setText(paraInfoList.get(selectedRow).getRestriction());
15188     }
15189     });
15190 }
15191     return paraInfoJTable;
15192 }
15193 private JLabel getTestStrategyJLabel() {
15194     if(testStrategyJLabel == null) {
15195         testStrategyJLabel = new JLabel();
15196         testStrategyJLabel.setText("\u6d4b\u8bd5\u7b56\u7565");
15197         testStrategyJLabel.setBounds(12, 266, 53, 17);
15198     }
15199     return testStrategyJLabel;
15200 }
15201 private JComboBox getTestStrategyJComboBox() {
15202     if(testStrategyJComboBox == null) {
15203         ComboBoxModel testStrategyJComboBoxModel =
15204             new DefaultComboBoxModel(
15205                 new String[] { "Random Testing", "Dynamic Random
15206 Testing" });
15207         testStrategyJComboBox = new JComboBox();
15208         testStrategyJComboBox.setModel(testStrategyJComboBoxModel);
15209         testStrategyJComboBox.setBounds(24, 295, 116, 24);
15210         testStrategyJComboBox.setFont(new java.awt.Font("Times New Roman", 0, 12));
15211     }
15212     return testStrategyJComboBox;
15213 }
15214 private JButton getJButtonNext() {
15215     if(jButtonNext == null) {
15216         jButtonNext = new JButton();
15217         jButtonNext.setText("\u4e0b\u4e00\u6b65");
15218         jButtonNext.setBounds(392, 367, 80, 30);
15219         jButtonNext.setAction(getNextAction());
15220     }
15221     return jButtonNext;
15222 }
15223 private JButton getJButtonPre() {
15224     if(jButtonPre == null) {
15225         jButtonPre = new JButton();
15226         jButtonPre.setText("\u4e0a\u4e00\u6b65");
15227         jButtonPre.setBounds(238, 367, 80, 30);
15228     }
15229     return jButtonPre;
15230 }
15231 private AbstractAction getNextAction() {
15232     if(nextAction == null) {
15233         nextAction = new AbstractAction("\u4e0b\u4e00\u6b65", null) {
15234             public void actionPerformed(ActionEvent evt) {
15235                 setting.put("Operation", (String)operationComboBox.getSelectedItem());
```

```
15236         setting.put("Test Strategy",
15237 (String)testStrategyJComboBox.getSelectedItem());
15238         SwingUtilities.invokeLater(new Runnable() {
15239             public void run() {
15240                 TestSettingFrame inst = new TestSettingFrame(setting,
15241 selectedOperationInfo);
15242                 inst.setLocationRelativeTo(null);
15243                 inst.setVisible(true);
15244             }
15245         });
15246     }
15247 };
15248 }
15249 return nextAction;
15250 }
15251 }
15252 package ui;
15253 import java.awt.Component;
15254 import java.awt.GridLayout;
15255 import java.awt.event.ActionEvent;
15256 import java.awt.event.KeyAdapter;
15257 import java.awt.event.KeyEvent;
15258 import java.awt.event.MouseAdapter;
15259 import java.awt.event.MouseEvent;
15260 import java.io.File;
15261 import javax.swing.AbstractAction;
15262 import javax.swing.ButtonGroup;
15263 import javax.swing.ComboBoxModel;
15264 import javax.swing.DefaultComboBoxModel;
15265 import javax.swing.JButton;
15266 import javax.swing.JComboBox;
15267 import javax.swing.JFileChooser;
15268 import javax.swing.JLabel;
15269 import javax.swing.JPanel;
15270 import javax.swing.JRadioButton;
15271 import javax.swing.JScrollPane;
15272 import javax.swing.JTable;
15273 import javax.swing.JTextField;
15274 import javax.swing.WindowConstants;
15275 import javax.swing.event.ChangeEvent;
15276 import javax.swing.event.ChangeListener;
15277 import javax.swing.SwingUtilities;
15278 import javax.swing.table.DefaultTableModel;
15279 import javax.swing.table.TableColumn;
15280 import cn.edu.ustb.webservicetester.model.OperationInfo;
15281 import cn.edu.ustb.webservicetester.model.ParameterInfo;
15282 import cn.edu.ustb.webservicetester.model.TestCase;
15283 import cn.edu.ustb.webservicetester.model.TestSet;
15284 import cn.edu.ustb.webservicetester.util.TestSetFileParser;
15285 import test.TestJFileChooser.MyFileFilter;
```

```
15286 public class TestSettingFrame extends javax.swing.JFrame {
15287     private JPanel jPanel1;
15288     private JButton nextPageJButton;
15289     private JRadioButton addTestCaseAutoRadioButton;
15290     private AbstractAction importTestCaseAction;
15291     private JButton addTestCaseAutoAdvancedSettingJButton;
15292     private JButton addTestCaseAutoBeginJButton;
15293     private JLabel addTestCaseAutoLabel2;
15294     private JTextField addTestCaseAutoTextField;
15295     private JLabel addTestCaseAutoLabel1;
15296     private JButton importTestCaseAddButton;
15297     private JButton importTestCaseScanButton;
15298     private JTextField importTestCaseTextField;
15299     private ButtonGroup buttonGroup1;
15300     private JRadioButton addTestCaseImportJRadioButton;
15301     private JLabel jLabel1;
15302     private JButton prePageButton;
15303     private JComboBox selectPageComboBox;
15304     private JScrollPane testSetDemoJScrollPane;
15305     private JTable testSetDemoJTable;
15306     private DefaultTableModel testSetDemoTableModel;
15307     private java.util.List<ParameterInfo> paraInfoList = null;
15308     private AbstractAction abstractAction1;
15309     private TestSet testSet = null;
15310     private OperationInfo operationInfo;
15311     private java.util.Map<String, String> testSetting = null;
15312     public static void main(String[] args) {
15313         SwingUtilities.invokeLater(new Runnable() {
15314             public void run() {
15315                 TestSettingFrame inst = new TestSettingFrame();
15316                 inst.setLocationRelativeTo(null);
15317                 inst.setVisible(true);
15318             }
15319         });
15320     }
15321     public TestSettingFrame() {
15322         super();
15323         initGUI();
15324     }
15325     public TestSettingFrame(java.util.Map<String, String> testSettingMap, OperationInfo
15326 operInfo) {
15327         super();
15328         testSetting = testSettingMap;
15329         initGUI(operInfo);
15330     }
15331     private void initGUI() {
15332         initGUI(null);
15333     }
15334     private void initGUI(OperationInfo oi) {
15335         operationInfo = oi;
```

```
15336     paraInfoList = operationInfo.getParaList();
15337     try {
15338         setTitle("测试配置");
15339         setResizable(false);
15340         GridLayout thisLayout = new GridLayout(1, 1);
15341         getContentPane().setLayout(thisLayout);
15342         {
15343             jPanel1 = new JPanel();
15344             getContentPane().add(jPanel1);
15345             jPanel1.setLayout(null);
15346             jPanel1.setBackground(new java.awt.Color(228,236,243));
15347             {
15348                 testSetDemoJScrollPane = new JScrollPane();
15349                 jPanel1.add(testSetDemoJScrollPane);
15350                 jPanel1.add(getNextPageJButton());
15351                 jPanel1.add(getSelectPageComboBox());
15352                 jPanel1.add(getPrePageButton());
15353                 jPanel1.add(getJLabel1());
15354                 jPanel1.add(getAddTestCaseImportJRadioButton());
15355                 jPanel1.add(getAddTestCaseAutoRadioButton());
15356                 jPanel1.add(getImpotyTestCaseTextField());
15357                 jPanel1.add(getImportTestCaseScanButton());
15358                 jPanel1.add(getImportTestCaseAddButton());
15359                 jPanel1.add(getAddTestCaseAutoLabel1());
15360                 jPanel1.add(getAddTestCaseAutoTextField());
15361                 jPanel1.add(getAddTestCaseAutoLabel2());
15362                 jPanel1.add(getAddTestCaseAutoBeginJButton());
15363                 jPanel1.add(getAddTestCaseAutoAdvancedSettingJButton());
15364                 getButtonGroup1();
15365                 addTestCaseImportJRadioButton.setSelected(true);
15366                 testSetDemoJScrollPane.setBounds(12, 12, 470, 183);
15367                 testSetDemoJScrollPane.setViewportView(getTestSetDemoJTable());
15368             }
15369         }
15370         setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
15371         pack();
15372         setSize(500, 444);
15373     } catch (Exception e) {
15374         e.printStackTrace();
15375     }
15376 }
15377 private JTable getTestSetDemoJTable() {
15378     if (testSetDemoJTable == null) {
15379         java.util.List<String> nl = new java.util.LinkedList<String>();
15380         nl.add("id");
15381         for (ParameterInfo pi : paraInfoList) {
15382             nl.add(pi.getName());
15383         }
15384         nl.add("expected result");
15385         String[] name = new String[nl.size()];
```



```
15386         nl.toArray(name);
15387         String[] a = {" "};
15388         String[][] data = {a};
15389         testSetDemoTableModel = new DefaultTableModel(data, name);
15390         testSetDemoJTable = new JTable(testSetDemoTableModel) {
15391             public boolean isCellEditable(int row, int column) {
15392                 return false;
15393             }
15394         };
15395         int w = testSetDemoJTable.getTableHeader().getWidth();
15396         testSetDemoJTable.getTableHeader().setSize(w, 17);
15397         testSetDemoJTable.setRowHeight(17);
15398     }
15399     return testSetDemoJTable;
15400 }
15401 private JButton getNextPageJButton() {
15402     if(nextPageJButton == null) {
15403         nextPageJButton = new JButton();
15404         nextPageJButton.setText("\u4e0b\u4e00\u9875");
15405         nextPageJButton.setBounds(393, 211, 90, 24);
15406         nextPageJButton.setAction(getAbstractAction1());
15407     }
15408     return nextPageJButton;
15409 }
15410 private JComboBox getSelectPageComboBox() {
15411     if(selectPageComboBox == null) {
15412         ComboBoxModel selectPageComboBoxModel =
15413             new DefaultComboBoxModel(
15414                 new String[] { "第 1 页", "第 2 页" });
15415         selectPageComboBox = new JComboBox();
15416         selectPageComboBox.setModel(selectPageComboBoxModel);
15417         selectPageComboBox.setBounds(279, 211, 88, 24);
15418         selectPageComboBox.setEditable(true);
15419     }
15420     return selectPageComboBox;
15421 }
15422 private JButton getPrePageButton() {
15423     if(prePageButton == null) {
15424         prePageButton = new JButton();
15425         prePageButton.setText("\u4e0a\u4e00\u9875");
15426         prePageButton.setBounds(164, 211, 90, 24);
15427     }
15428     return prePageButton;
15429 }
15430 private JLabel getJLabel1() {
15431     if(jLabel1 == null) {
15432         jLabel1 = new JLabel();
15433         jLabel1.setText("\u5171 0 \u6761\u6d4b\u8bd5\u7528\u4f8b");
15434         jLabel1.setBounds(12, 205, 134, 21);
15435         jLabel1.setForeground(new java.awt.Color(255,0,0));
```

```
15436     }
15437     return jLabel1;
15438 }
15439 private JRadioButton getAddTestCaseImportJRadioButton() {
15440     if(addTestCaseImportJRadioButton == null) {
15441         addTestCaseImportJRadioButton = new JRadioButton();
15442         addTestCaseImportJRadioButton.setText("\u5bfc\u5165\u6587\u4ef6");
15443         addTestCaseImportJRadioButton.setBounds(12, 275, 80, 21);
15444         addTestCaseImportJRadioButton.setBackground(new
15445 java.awt.Color(252,252,203));
15446         addTestCaseImportJRadioButton.addChangeListener(new ChangeListener() {
15447             public void stateChanged(ChangeEvent evt) {
15448                 addTestCaseImportJRadioButtonStateChanged(evt);
15449             }
15450         });
15451     }
15452     return addTestCaseImportJRadioButton;
15453 }
15454 private JRadioButton getAddTestCaseAutoRadioButton() {
15455     if(addTestCaseAutoRadioButton == null) {
15456         addTestCaseAutoRadioButton = new JRadioButton();
15457         addTestCaseAutoRadioButton.setText("\u81ea\u52a8\u6dfb\u52a0");
15458         addTestCaseAutoRadioButton.setBounds(12, 345, 80, 21);
15459         addTestCaseAutoRadioButton.setBackground(new java.awt.Color(252,252,203));
15460     }
15461     return addTestCaseAutoRadioButton;
15462 }
15463 private ButtonGroup getButtonGroup1() {
15464     if(buttonGroup1 == null) {
15465         buttonGroup1 = new ButtonGroup();
15466     }
15467     buttonGroup1.add(addTestCaseImportJRadioButton);
15468     buttonGroup1.add(addTestCaseAutoRadioButton);
15469     return buttonGroup1;
15470 }
15471 private JTextField getImpotyTestCaseTextField() {
15472     if(importTestCaseTextField == null) {
15473         importTestCaseTextField = new JTextField();
15474         importTestCaseTextField.setBounds(97, 274, 206, 24);
15475     }
15476     return importTestCaseTextField;
15477 }
15478 private JButton getImportTestCaseScanButton() {
15479     if(importTestCaseScanButton == null) {
15480         importTestCaseScanButton = new JButton();
15481         importTestCaseScanButton.setText("\u6d4f\u89c8");
15482         importTestCaseScanButton.setBounds(303, 274, 54, 24);
15483         importTestCaseScanButton.setFont(new java.awt.Font("微软雅黑",0,10));
15484         final javax.swing.JFrame parentFrame = this;
15485         importTestCaseScanButton.addMouseListener(new MouseAdapter() {
```

```

15486         public void mouseClicked(MouseEvent evt) {
15487             JFileChooser jf = new JFileChooser();
15488             jf.setApproveButtonText("导入");
15489             jf.setCurrentDirectory(new File("."));
15490             jf.setMultiSelectionEnabled(false);
15491             jf.setFileFilter(new MyFileFilter("xml"));
15492             jf.setDialogTitle("请选择要导入的测试集文件");
15493             jf.showOpenDialog(parentFrame);
15494             File f = jf.getSelectedFile();
15495             if (f != null) {
15496                 importTestCaseTextField.setText(f.getPath());
15497             }
15498         }
15499     });
15500 }
15501 return importTestCaseScanButton;
15502 }
15503 private JButton getImportTestCaseAddButton() {
15504     if(importTestCaseAddButton == null) {
15505         importTestCaseAddButton = new JButton();
15506         importTestCaseAddButton.setText("\u5bfc\u5165");
15507         importTestCaseAddButton.setBounds(360, 274, 54, 24);
15508         importTestCaseAddButton.setFont(new java.awt.Font("微软雅黑",0,10));
15509         importTestCaseAddButton.setAction(getImportTestCaseAction());
15510     }
15511     return importTestCaseAddButton;
15512 }
15513 private JLabel getAddTestCaseAutoLabel1() {
15514     if(addTestCaseAutoLabel1 == null) {
15515         addTestCaseAutoLabel1 = new JLabel();
15516         addTestCaseAutoLabel1.setText("\u6dfb\u52a0");
15517         addTestCaseAutoLabel1.setBounds(97, 347, 30, 17);
15518     }
15519     return addTestCaseAutoLabel1;
15520 }
15521 private JTextField getAddTestCaseAutoTextField() {
15522     if(addTestCaseAutoTextField == null) {
15523         addTestCaseAutoTextField = new JTextField();
15524         addTestCaseAutoTextField.setBounds(133, 344, 56, 24);
15525         addTestCaseAutoTextField.setToolTipText("\u81ea\u52a8\u751f\u6210\u6d4b\u8bd5\u7528\u4f8b\u7684\u6570\u91cf");
15526         addTestCaseAutoTextField.setHorizontalAlignment(JTextField.RIGHT);
15527         addTestCaseAutoTextField.addKeyListener(new KeyAdapter() {
15528             public void keyTyped(KeyEvent evt) {
15529                 char kc = evt.getKeyChar();
15530                 if (addTestCaseAutoTextField.getText().length() == 0 && kc >=
15531                     KeyEvent.VK_1 && kc <= KeyEvent.VK_9) {
15532                     ;
15533                 } else if (addTestCaseAutoTextField.getText().length() > 0 &&
15534                     addTestCaseAutoTextField.getText().length() < 6 && kc >= KeyEvent.VK_0 && kc <=

```

```

15536 KeyEvent.VK_9) {
15537             } else {
15538                 evt.consume();
15539             }
15540         }
15541     });
15542     addTestCaseAutoTextField.setEnabled(false);
15543 }
15544 return addTestCaseAutoTextField;
15545 }
15546 private JLabel getAddTestCaseAutoLabel2() {
15547     if(addTestCaseAutoLabel2 == null) {
15548         addTestCaseAutoLabel2 = new JLabel();
15549         addTestCaseAutoLabel2.setText("\u6761\u6d4b\u8bd5\u7528\u4f8b");
15550         addTestCaseAutoLabel2.setBounds(197, 347, 70, 17);
15551     }
15552     return addTestCaseAutoLabel2;
15553 }
15554 private JButton getAddTestCaseAutoBeginJButton() {
15555     if(addTestCaseAutoBeginJButton == null) {
15556         addTestCaseAutoBeginJButton = new JButton();
15557         addTestCaseAutoBeginJButton.setText("\u5f00\u59cb");
15558         addTestCaseAutoBeginJButton.setBounds(290, 345, 90, 24);
15559         addTestCaseAutoBeginJButton.setEnabled(false);
15560     }
15561     return addTestCaseAutoBeginJButton;
15562 }
15563 private JButton getAddTestCaseAutoAdvancedSettingJButton() {
15564     if(addTestCaseAutoAdvancedSettingJButton == null) {
15565         addTestCaseAutoAdvancedSettingJButton = new JButton();
15566         addTestCaseAutoAdvancedSettingJButton.setText("\u9ad8\u7ea7\u8bbe\u7f6e");
15567         addTestCaseAutoAdvancedSettingJButton.setBounds(393, 345, 90, 24);
15568         addTestCaseAutoAdvancedSettingJButton.setFont(new java.awt.Font("微软雅黑",0,10));
15569         addTestCaseAutoAdvancedSettingJButton.setEnabled(false);
15570     }
15571 }
15572 return addTestCaseAutoAdvancedSettingJButton;
15573 }
15574 private AbstractAction getImportTestCaseAction() {
15575     if(importTestCaseAction == null) {
15576         importTestCaseAction = new AbstractAction("\u5bfc\u5165", null) {
15577             public void actionPerformed(ActionEvent evt) {
15578                 File f = new File(importTestCaseTextField.getText());
15579                 TestSetFileParser tsfp = new TestSetFileParser();
15580                 tsfp.setParaInfoList(paraInfoList);
15581                 testSet = tsfp.parseFile(importTestCaseTextField.getText());
15582                 for (int i = 0; i < testSet.numOfTestCases(); ++i) {
15583                     TestCase tc = testSet.get(i);
15584                     java.util.List<String> pValueList = new
15585 java.util.ArrayList<String>(paraInfoList.size() + 2);

```

```

15586         pValueList.add(new Integer(tc.getId()).toString());
15587         for (ParameterInfo pi : paraInfoList) {
15588             pValueList.add(tc.getValueOfParameter(pi));
15589         }
15590         pValueList.add(tc.getExpectedResult());
15591         String[] pValueArray = new String[pValueList.size()];
15592         pValueList.toArray(pValueArray);
15593         for (String pv : pValueArray) {
15594             System.out.println(pv);
15595         }
15596         testSetDemoTableModel.addRow(pValueArray);
15597     }
15598 }
15599 };
15600 }
15601 return importTestCaseAction;
15602 }
15603 private AbstractAction getAbstractAction1() {
15604     if (abstractAction1 == null) {
15605         abstractAction1 = new AbstractAction("\u4e0b\u4e00\u9875", null) {
15606             public void actionPerformed(ActionEvent evt) {
15607                 if (testSet != null) {
15608                     String testStrategy = testSetting.get("Test Strategy");
15609                     testSetting.put("Test Set", importTestCaseTextField.getText());
15610                     System.out.println(testStrategy);
15611                     if ("Random Testing".equals(testStrategy)) {
15612                         ExecutionFrame ef = new ExecutionFrame(testSetting,
15613 operationInfo, testSet);
15614                         ef.setLocationRelativeTo(null);
15615                         ef.setVisible(true);
15616                     } else if ("Dynamic Random Testing".equals(testStrategy)){
15617                         System.out.println("??");
15618                         PartitionSettingFrame2 psf = new
15619 PartitionSettingFrame2(testSetting, operationInfo, testSet);
15620                         psf.setLocationRelativeTo(null);
15621                         psf.setVisible(true);
15622                     } else {
15623                         System.out.println("??");
15624                     }
15625                 } else {
15626                 }
15627             }
15628         };
15629     }
15630     return abstractAction1;
15631 }
15632 private void addTestCaseImportJRadioButtonStateChanged(ChangeEvent evt) {
15633     System.out.println("addTestCaseImportJRadioButton.stateChanged, event="+evt);
15634     if (addTestCaseImportJRadioButton.isSelected()) {
15635         importTestCaseTextField.setEnabled(true);

```

```
15636         importTestCaseScanButton.setEnabled(true);
15637         importTestCaseAddButton.setEnabled(true);
15638         addTestCaseAutoTextField.setEnabled(false);
15639         addTestCaseAutoBeginJButton.setEnabled(false);
15640         addTestCaseAutoAdvancedSettingJButton.setEnabled(false);
15641     } else {
15642         importTestCaseTextField.setEnabled(false);
15643         importTestCaseScanButton.setEnabled(false);
15644         importTestCaseAddButton.setEnabled(false);
15645         addTestCaseAutoTextField.setEnabled(true);
15646         addTestCaseAutoBeginJButton.setEnabled(true);
15647         addTestCaseAutoAdvancedSettingJButton.setEnabled(true);
15648     }
15649 }
15650 }
15651 package test;
15652 import java.io.File;
15653 import javax.swing.JFileChooser;
15654 import javax.swing.filechooser.FileFilter;
15655 public class TestJFileChooseer {
15656     public static void main(String[] args) {
15657         JFileChooser jf = new JFileChooser();
15658         jf.setApproveButtonText("导入");
15659         jf.setCurrentDirectory(new File("."));
15660         jf.setMultiSelectionEnabled(false);
15661         jf.setFileFilter(new MyFileFilter("xml"));
15662         jf.setDialogTitle("请选择要导入的测试集文件");
15663         jf.showOpenDialog(null);
15664         File f = jf.getSelectedFile();
15665         if (f != null)
15666             System.out.println(f.getPath());
15667     }
15668     public static class MyFileFilter extends FileFilter {
15669         private String ext;
15670         public MyFileFilter(String extString) {
15671             ext = extString;
15672         }
15673         public boolean accept(File f) {
15674             if (f.isDirectory()) {
15675                 return true;
15676             }
15677             String extension = getExtension(f);
15678             if (extension.toLowerCase().equals(this.ext.toLowerCase()))
15679                 {
15680                     return true;
15681                 }
15682             return false;
15683         }
15684         public String getDescription() {
15685             return this.ext.toUpperCase();
15686         }
15687     }
15688 }
```

```
15686     }
15687     private String getExtension(File f) {
15688         String name = f.getName();
15689         int index = name.lastIndexOf('.');
15690         if (index == -1)
15691         {
15692             return "";
15693         }
15694         else
15695         {
15696             return name.substring(index + 1).toLowerCase();
15697         }
15698     }
15699 }
15700 }
15701 package test;
15702 import java.awt.BorderLayout;
15703 import javax.swing.DefaultCellEditor;
15704 import javax.swing.JPanel;
15705 import javax.swing.JScrollPane;
15706 import javax.swing.JTable;
15707 import javax.swing.JTextField;
15708 import javax.swing.WindowConstants;
15709 import javax.swing.SwingUtilities;
15710 public class TestJTableFrame extends javax.swing.JFrame {
15711     private JPanel jPanel1;
15712     private JScrollPane jScrollPane1;
15713     private JTable jTable1;
15714     public static void main(String[] args) {
15715         SwingUtilities.invokeLater(new Runnable() {
15716             public void run() {
15717                 TestJTableFrame inst = new TestJTableFrame();
15718                 inst.setLocationRelativeTo(null);
15719                 inst.setVisible(true);
15720             }
15721         });
15722     }
15723     public TestJTableFrame() {
15724         super();
15725         initGUI();
15726     }
15727     private void initGUI() {
15728         try {
15729             setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
15730             {
15731                 jPanel1 = new JPanel();
15732                 jPanel1.setLayout(null);
15733                 getContentPane().add(jPanel1, BorderLayout.CENTER);
15734                 {
15735                     jScrollPane1 = new JScrollPane();
```

```
15736         jPanel1.add(jScrollPane1);
15737         jScrollPane1.setBounds(29, 36, 195, 136);
15738         jScrollPane1.setViewportView(getJTable1());
15739     }
15740 }
15741 pack();
15742 setSize(400, 300);
15743 jTable1.editCellAt(0, 0);
15744 jTable1.getEditorComponent().requestFocusInWindow();
15745 } catch (Exception e) {
15746     e.printStackTrace();
15747 }
15748 }
15749 private JTable getJTable1() {
15750     if (jTable1 == null) {
15751         String[] s1 = new String[] {"小王", "0001"};
15752         String[] s2 = new String[] {"小芳", "0002"};
15753         String[][] data = new String[][] {s1, s2};
15754         String[] name = new String[] {"姓名", "工号"};
15755         jTable1 = new JTable(data, name);
15756     }
15757     return jTable1;
15758 }
15759 }
15760 package test;
15761 import java.io.File;
15762 import org.dom4j.Document;
15763 import org.dom4j.DocumentException;
15764 import org.dom4j.Element;
15765 public class XmlTest {
15766     public static void main(String[] args) {
15767         File f = new File("E:\\研究生\\毕业设计\\DRT 算法改进\\实验\\停车计费\\测试用例
15768 集\\随机 1000 规模\\testcaseset4RT.xml");
15769         org.dom4j.io.SAXReader saxReader = new org.dom4j.io.SAXReader();
15770         Document document = null;
15771         Element rootElement = null;
15772         try {
15773             document = saxReader.read(f);
15774             rootElement = document.getRootElement();
15775             java.util.List<Element> testCaseList = rootElement.selectNodes("testCase");
15776             System.out.println(testCaseList.size());
15777         } catch (DocumentException e) {
15778             e.printStackTrace();
15779         }
15780     }
15781 }
15782 package test;
15783 import cn.edu.ustb.mt4ws.tcg.WsdlOperationFormat;
15784 import cn.edu.ustb.mt4ws.wsdl.parser.WsdlReader11;
15785 import javax.wsdl.*;
```



```

15786 import javax.wsdl.extensions.*;
15787 import javax.wsdl.factory.*;
15788 import javax.wsdl.xml.*;
15789 import javax.xml.namespace.QName;
15790 import java.io.*;
15791 import java.util.*;
15792 import org.dom4j.DocumentException;
15793 import org.dom4j.io.SAXReader;
15794 import org.w3c.dom.*;
15795 public class ZzqTest {
15796     public static void main(String[] args) {
15797         String wsUrl = "http://localhost:8080/axis2/services/ATMService?wsdl";
15798         File f = new File("temp.xml");
15799         if (!f.exists()) {
15800             try {
15801                 WSDLFactory factory = WSDLFactory.newInstance();
15802                 WSDLReader reader = factory.newWSDLReader();
15803                 reader.setFeature("javax.wsdl.verbose", true);
15804                 reader.setFeature("javax.wsdl.importDocuments", true);
15805                 Definition def = reader.readWSDL(wsUrl);
15806                 Writer xmlWriter = new BufferedWriter(new OutputStreamWriter(new
15807 FileOutputStream("temp.xml")));
15808                 WSDLWriter wsdlWriter = factory.newWSDLWriter();
15809                 wsdlWriter.writeWSDL(def, xmlWriter);
15810             } catch (FileNotFoundException e) {
15811                 e.printStackTrace();
15812             } catch (WSDLException e) {
15813                 e.printStackTrace();
15814             }
15815         }
15816         SAXReader dReader = new SAXReader();
15817         org.dom4j.Document document;
15818         try {
15819             document = dReader.read(new File("temp.xml"));
15820             org.dom4j.Element root = document.getRootElement();
15821             org.dom4j.Element typesElement = root.element("types");
15822             org.dom4j.Element schemaElement = typesElement.element("schema");
15823             @SuppressWarnings("unchecked")
15824             List<org.dom4j.Element> elist =
15825 (List<org.dom4j.Element>)schemaElement.elements();
15826             for (org.dom4j.Element e : elist) {
15827                 if (e.attribute("name").getValue().equals("accountNumber")) {
15828                     org.dom4j.Element rElement = e.element("restriction");
15829                     String nativeType = rElement.attribute("base").getValue();
15830                     System.out.println(nativeType);
15831                 }
15832             }
15833         } catch (DocumentException e) {
15834             e.printStackTrace();
15835         }

```

```
15836     }
15837 }
15838 package test.thread;
15839 import java.util.*;
15840 import java.util.concurrent.TimeUnit;
15841 public class ThreadTestMain {
15842     static class WriteString implements Runnable {
15843         private List<String> sl;
15844         public WriteString(List<String> ssl) {
15845             sl = ssl;
15846         }
15847         public void run() {
15848             int i = 0;
15849             String s;
15850             while (i++ < 20) {
15851                 s = new Integer(i).toString();
15852                 sl.add(s);
15853                 try {
15854                     TimeUnit.MILLISECONDS.sleep(new Random().nextInt(1500));
15855                 } catch (InterruptedException e) {
15856                     e.printStackTrace();
15857                     System.out.println("size:" + sl.size());
15858                     System.out.print(i + ": ");
15859                     System.out.println(sl.get(i));
15860                 }
15861             }
15862         }
15863     }
15864 }
15865 static class ReadString implements Runnable {
15866     List<String> sl;
15867     public ReadString(List<String> ssl) {
15868         sl = ssl;
15869     }
15870     public void run() {
15871         int i = 0;
15872         while (i < 20) {
15873             if (i < sl.size()) {
15874                 System.out.println("size:" + sl.size());
15875                 System.out.print(i + ": ");
15876                 System.out.println(sl.get(i));
15877                 ++i;
15878             } else {
15879                 try {
15880                     TimeUnit.MILLISECONDS.sleep(2000);
15881                 } catch (InterruptedException e) {
15882                     e.printStackTrace();
15883                 }
15884             }
15885         }
15886     }
15887 }
```

```
15886     }
15887 }
15888 public static void main(String[] args) {
15889     List<String> sl = new ArrayList<String>();
15890     Thread th1 = new Thread(new WriteString(sl));
15891     Thread th2 = new Thread(new ReadString(sl));
15892     th1.start();
15893     th2.start();
15894 }
15895 }
15896 package cn.edu.ustb.webservicetester.controller;
15897 import javax.swing.SwingUtilities;
15898 import ui.OperationSelectFrame;
15899 import ui.OperationSelectionFrame2;
15900 import ui.PreferenceFrame;
15901 import cn.edu.ustb.webservicetester.util.WsdlParser;
15902 public class ParseWsdlControllor {
15903     public void parseWsdlControl(String wsdlUrl) {
15904         WsdlParser parser = new WsdlParser();
15905         final java.util.List opInfoList = parser.parseWsdl(wsdlUrl);
15906         final java.util.Map<String, String> setting = new java.util.HashMap<String, String>();
15907         setting.put("Web Service", wsdlUrl.substring(0, wsdlUrl.lastIndexOf("?wsdl")));
15908         SwingUtilities.invokeLater(new Runnable() {
15909             public void run() {
15910                 PreferenceFrame inst = new PreferenceFrame(setting, opInfoList);
15911                 inst.setLocationRelativeTo(null);
15912                 inst.setVisible(true);
15913             }
15914         });
15915     }
15916 }
15917 package cn.edu.ustb.webservicetester.model;
15918 public class ParameterIntervalScenario {
15919     private String range;// 区间范围
15920     private int weight;// 权重
15921     public String getRange() {
15922         return range;
15923     }
15924     public void setRange(String range) {
15925         this.range = range;
15926     }
15927     public int getWeight() {
15928         return weight;
15929     }
15930     public void setWeight(int weight) {
15931         this.weight = weight;
15932     }
15933 }
15934 package cn.edu.ustb.webservicetester.model;
15935 public class ParameterPartitionScenario {
```

```
15936     private ParameterInfo paraInfo;// 参数信息
15937     private java.util.List<ParameterIntervalScenario> partitionScenario;// 分区方案
15938     public ParameterPartitionScenario() {
15939         partitionScenario = new java.util.LinkedList<ParameterIntervalScenario>();
15940     }
15941     public ParameterInfo getParaInfo() {
15942         return paraInfo;
15943     }
15944     public void setParaInfo(ParameterInfo paraInfo) {
15945         this.paraInfo = paraInfo;
15946     }
15947     public ParameterIntervalScenario get(int index) {
15948         return partitionScenario.get(index);
15949     }
15950     public void add(ParameterIntervalScenario paraInter) {
15951         partitionScenario.add(paraInter);
15952     }
15953     public int numOfIntervals() {
15954         return partitionScenario.size();
15955     }
15956 }
15957 package cn.edu.ustb.webservicetester.model;
15958 import java.util.Map.Entry;
15959 public class Partition extends TestSet {
15960     private java.util.Map<ParameterInfo, String> rules;
15961     public Partition() {
15962         super();
15963         rules = new java.util.HashMap<ParameterInfo, String>();
15964     }
15965     public Partition(java.util.Map<ParameterInfo, String> rs) {
15966         super();
15967         rules = rs;
15968     }
15969     public void addRule(ParameterInfo pi, String rStr) {
15970         rules.put(pi, rStr);
15971     }
15972     public String getRule(ParameterInfo pi) {
15973         if (!rules.containsKey(pi))
15974             return null;
15975         return rules.get(pi);
15976     }
15977     public boolean couldAdd(TestCase tc) {
15978         java.util.List<ParameterInfo> paraInfoList = tc.getParameters();
15979         boolean ca = true;
15980         for (int i = 0; i < paraInfoList.size() && ca; ++i) {
15981             ParameterInfo pi = paraInfoList.get(i);
15982             if (pi.getType() == ParameterInfo.WHATEVER) {
15983                 continue;
15984             }
15985             if (!rules.containsKey(pi)) {
```

```
15986         ca = false;
15987         continue;
15988     }
15989     String ci = pi.getClassInfo();
15990     if (pi.getType() == ParameterInfo.SCATTERED) {
15991         ca = tc.getValueOfParameter(pi).equals(rules.get(pi));
15992     } else {
15993         String rule = rules.get(pi);
15994         String ss[] = rule.split(",");
15995         String s1 = ss[0].substring(1).trim();
15996         String s2 = ss[1].substring(0, ss[1].length() - 1).trim();
15997         String s = tc.getValueOfParameter(pi);
15998         if ("byte".equals(ci)
15999             || "short".equals(ci)
16000             || "int".equals(ci)
16001             || "long".equals(ci)) {
16002             long l1 = new Long(s1);
16003             long l2 = new Long(s2);
16004             long l = new Long(s);
16005             if (l < l2 && l > l1) {
16006                 ca = true;
16007             } else if (rule.startsWith("[") && l == l1) {
16008                 ca = true;
16009             } else if (rule.endsWith("]") && l == l2) {
16010                 ca = true;
16011             } else {
16012                 ca = false;
16013             }
16014         } else if ("float".equals(ci) || "double".equals(ci)) {
16015             double d1 = new Double(s1);
16016             double d2 = new Double(s2);
16017             System.out.println(pi.getName());
16018             System.out.println(s);
16019             System.out.println(tc.containsParameter(pi));
16020             System.out.println(i);
16021             rsb.append(ent.getKey());
16022             rsb.append(": ");
16023             rsb.append(ent.getValue());
16024             rsb.append("\n");
16025             double d = new Double(s);
16026             if (d < d2 && d > d1) {
16027                 ca = true;
16028             } else if (rule.startsWith("[") && Math.abs(d - d1) < 1e-5) {
16029                 ca = true;
16030             } else if (rule.endsWith("]") && Math.abs(d2 - d) < 1e-5) {
16031                 ca = true;
16032             } else {
16033                 ca = false;
16034             }
16035         } else if ("char".equals(ci)
```

```
16036         && s1.length() == 1
16037         && s2.length() == 1
16038         && s.length() == 1) {
16039             char c1 = s1.charAt(0);
16040             char c2 = s2.charAt(0);
16041             char c = s.charAt(0);
16042             if (c < c2 && c > c1) {
16043                 ca = true;
16044             } else if (rule.startsWith("[") && c == c1) {
16045                 ca = true;
16046             } else if (rule.endsWith("]") && c == c2) {
16047                 ca = true;
16048             } else {
16049                 ca = false;
16050             }
16051         } else {
16052             }
16053     }
16054 }
16055 return ca;
16056 }
16057 public String toString() {
16058     StringBuffer rsb = new StringBuffer();
16059     for (Entry<ParameterInfo, String> ent : rules.entrySet()) {
16060         rsb.append(ent.getKey());
16061         rsb.append(": ");
16062         rsb.append(ent.getValue());
16063         rsb.append("\n");
16064     }
16065     rsb.append("[\n");
16066     java.util.Iterator<TestCase> tcIter = testCases.iterator();
16067     while (tcIter.hasNext()) {
16068         TestCase tc = tcIter.next();
16069         rsb.append(tc);
16070         rsb.append("\n");
16071         sb.append(p.getName());
16072         sb.append(" : ");
16073         sb.append(paraValues.get(p));
16074     }
16075     rsb.append("]");
16076     return rsb.toString();
16077 }
16078 }
16079 package cn.edu.ustb.webservicetester.model;
16080 public class TestCase {
16081     private int id = -1;
16082     private java.util.Map<ParameterInfo, String> paraValues;
16083     private String expectedResult = null;
16084     public TestCase() {
16085         paraValues = new java.util.HashMap<ParameterInfo, String>();
```

```
16086     }
16087     public boolean containsParameter(ParameterInfo pi) {
16088         return paraValues.containsKey(pi);
16089     }
16090     public int getId() {
16091         return id;
16092     }
16093     public void setId(int id) {
16094         this.id = id;
16095     }
16096     public void setValueOfParameter(ParameterInfo pi, String value) {
16097         paraValues.put(pi, value);
16098     }
16099     public String getValueOfParameter(ParameterInfo pi) {
16100         return paraValues.get(pi);
16101     }
16102     public String getExpectedResult() {
16103         return expectedResult;
16104     }
16105     public void setExpectedResult(String expectedResult) {
16106         this.expectedResult = expectedResult;
16107     }
16108     public java.util.List<ParameterInfo> getParameters() {
16109         java.util.List<ParameterInfo> paraInfoList = new
16110 java.util.LinkedList<ParameterInfo>();
16111         java.util.Iterator<ParameterInfo> pii = paraValues.keySet().iterator();
16112         while (pii.hasNext()) {
16113             paraInfoList.add(pii.next());
16114             sb.append(p.getName());
16115             sb.append(" : ");
16116             sb.append(paraValues.get(p));
16117             sb.append(p.getName());
16118             sb.append(" : ");
16119             sb.append(paraValues.get(p));
16120             sb.append(", ");
16121         }
16122         return paraInfoList;
16123     }
16124     public boolean equals(Object o) {
16125         if (!(o instanceof TestCase)) {
16126             return false;
16127         }
16128         TestCase tc = (TestCase)o;
16129         if (!((tc.getParameters()).equals(getParameters())) {
16130             return false;
16131         }
16132         java.util.Iterator<ParameterInfo> paraInfoIter = paraValues.keySet().iterator();
16133         boolean findDifference = false;
16134         while (paraInfoIter.hasNext() && !findDifference) {
```

```
16136         ParameterInfo pi = paraInfoIter.next();
16137         findDifference = !(paraValues.get(pi).equals(tc.getValueOfParameter(pi)));
16138         sb.append(p.getName());
16139         sb.append(" : ");
16140         sb.append(paraValues.get(p));
16141         sb.append(", ");
16142     }
16143     return !findDifference;
16144 }
16145 public int hashCode() {
16146     int result = 17;
16147     result = result * 37 + paraValues.hashCode();
16148     if (expectedResult != null)
16149         result = result * 37 + expectedResult.hashCode();
16150     return result;
16151 }
16152 public String toString() {
16153     StringBuffer sb = new StringBuffer();
16154     for (ParameterInfo p : getParameters()) {
16155         sb.append(p.getName());
16156         sb.append(" : ");
16157         sb.append(paraValues.get(p));
16158         sb.append(", ");
16159     }
16160     if (expectedResult != null) {
16161         sb.append("expected result : ");
16162         sb.append(expectedResult);
16163     } else {
16164         sb.delete(sb.length() - 2, sb.length());
16165     }
16166     return sb.toString();
16167 }
16168 }
16169 package cn.edu.ustb.webservicetester.model;
16170 public class TestSet {
16171     protected java.util.List<TestCase> testCases;
16172     public TestSet() {
16173         testCases = new java.util.LinkedList<TestCase>();
16174     }
16175     public void add(TestCase tc) {
16176         testCases.add(tc);
16177     }
16178     public TestCase get(int i) {
16179         return testCases.get(i);
16180     }
16181     public boolean contains(TestCase tc) {
16182         boolean result = false;
16183         java.util.Iterator<TestCase> tsIter = testCases.iterator();
16184         while (tsIter.hasNext() && !result) {
16185             result = tc.equals(tsIter.next());
```



```
16186     }
16187     return false;
16188 }
16189 public int numOfTestCases() {
16190     return testCases.size();
16191 }
16192 }
16193 package cn.edu.ustb.webservicetester.model;
16194 public class TestSetWithPartition {
16195     private Partition[] partitions;
16196     private double[] profile;
16197     public TestSetWithPartition() {
16198         partitions = null;
16199         profile = null;
16200     }
16201     public TestSetWithPartition(int numOfPartitions) {
16202         partitions = new Partition[numOfPartitions];
16203         setProfileEvenly();
16204     }
16205     public TestSetWithPartition(Partition[] ps) {
16206         setPartitions(ps);
16207         setProfileEvenly();
16208     }
16209     public TestSetWithPartition(Partition[] ps, double[] prf) {
16210         setPartitions(ps);
16211         setProfile(prf);
16212     }
16213     public void setPartitions(Partition[] ps) {
16214         if (ps != null && ps.length > 0) {
16215             int i = 0;
16216             for (Partition p : ps) {
16217                 if (p.numOfTestCases() > 0) {
16218                     ++i;
16219                 }
16220             }
16221             partitions = new Partition[i];
16222             i = 0;
16223             for (int j = 0; j < ps.length; ++j) {
16224                 if (ps[j].numOfTestCases() > 0) {
16225                     partitions[i] = ps[j];
16226                     ++i;
16227                 }
16228             }
16229         }
16230     }
16231     public void setProfile(double[] prf) {
16232         if (prf != null && prf.length > 0) {
16233             profile = new double[prf.length];
16234             for (int i = 0; i < prf.length; ++i) {
16235                 profile[i] = prf[i];
```

```
16236         }
16237     }
16238 }
16239 public void setProfileEvenly() {
16240     if (partitions != null && partitions.length > 0) {
16241         profile = new double[partitions.length];
16242         for (int i = 0; i < profile.length; ++i) {
16243             profile[i] = 1.0 / profile.length;
16244         }
16245     }
16246 }
16247 public int numOfPartitions() {
16248     if (partitions == null)
16249         return 0;
16250     return partitions.length;
16251 }
16252 public int numOfTestCases() {
16253     int n = 0;
16254     if (partitions != null && partitions.length > 0) {
16255         for (Partition p : partitions) {
16256             n += p.numOfTestCases();
16257         }
16258     }
16259     return n;
16260 }
16261 public Partition[] getPartitions() {
16262     return partitions;
16263 }
16264 public Partition getPartition(int i) {
16265     return partitions[i];
16266 }
16267 public TestCase getTestCase(int pIndex, int tcIndex) {
16268     Partition p = partitions[pIndex];
16269     return p.get(tcIndex);
16270 }
16271 public double[] getProfile() {
16272     return profile;
16273 }
16274 }
16275 package cn.edu.ustb.webservicetester.model;
16276 public class Interval {
16277     private java.util.Map<ParameterInfo, ParameterIntervalScenario> range;
16278     private int weight;
16279     private java.util.List<TestCase> testCases;
16280     public Interval(java.util.Map<ParameterInfo, ParameterIntervalScenario> r, int w) {
16281         this.range = r;
16282         testCases = new java.util.LinkedList<TestCase>();
16283     }
16284     public int getWeight() {
16285         return weight;
```

```
16286     }
16287     public int numOfTestCases() {
16288         return testCases.size();
16289     }
16290     public void addTestCase(TestCase tc) {
16291         testCases.add(tc);
16292     }
16293     public TestCase getTestCase(int i) {
16294         return testCases.get(i);
16295     }
16296     public boolean includes(TestCase tc) {
16297         for (ParameterInfo pi : tc.getParameters()) {
16298             if (pi.getType() == ParameterInfo.SCATTERED) {
16299                 if (excludeScatteredValue(tc, pi))
16300                     return false;
16301             } else if (pi.getType() == ParameterInfo.CONTINUOUS){
16302                 if (excludesContinuousValue(tc, pi))
16303                     return false;
16304             }
16305         }
16306         return true;
16307     }
16308     private boolean excludesContinuousValue(TestCase tc, ParameterInfo pi) {
16309         String rangeOfParaInterval = range.get(pi).getRange();
16310         String valueOfTestCase = tc.getValueOfParameter(pi);
16311         double upperBound;
16312         double lowerBound;
16313         String[] ss = rangeOfParaInterval.split(", ");
16314         lowerBound = Double.valueOf(ss[0].substring(1));
16315         upperBound = Double.valueOf(ss[1].substring(0, ss[1].length() - 1));
16316         double votc = Double.valueOf(valueOfTestCase);
16317         if (rangeOfParaInterval.startsWith("[") && votc < lowerBound)
16318             return true;
16319         if (rangeOfParaInterval.startsWith("(") && votc <= lowerBound)
16320             return true;
16321         if (rangeOfParaInterval.endsWith("]") && votc > upperBound)
16322             return true;
16323         if (rangeOfParaInterval.endsWith(")") && votc >= upperBound)
16324             return true;
16325         return false;
16326     }
16327     private boolean excludeScatteredValue(TestCase tc, ParameterInfo pi) {
16328         return !tc.getValueOfParameter(pi).equals(range.get(pi.getName()).getRange());
16329     }
16330 }
16331 package cn.edu.ustb.webservicetester.model;
16332 import java.util.List;
16333 public class OperationInfo {
16334     private String opName;
16335     private List<ParameterInfo> paraList;
```

```
16336     public OperationInfo() {
16337     }
16338     public String getOpName() {
16339         return opName;
16340     }
16341     public void setOpName(String opName) {
16342         this.opName = opName;
16343     }
16344     public List<ParameterInfo> getParaList() {
16345         return paraList;
16346     }
16347     public void setParaList(List<ParameterInfo> paraList) {
16348         this.paraList = paraList;
16349     }
16350 }
16351 package cn.edu.ustb.webservicetester.model;
16352 public class ParameterInfo {
16353     public static final int WHATEVER = 0;
16354     public static final int CONTINUOUS = 1;
16355     public static final int SCATTERED = 2;
16356     private String name;
16357     private String classInfo;
16358     private int type;
16359     private String restriction;
16360     public ParameterInfo() {
16361         type = WHATEVER;
16362     }
16363     public String getName() {
16364         return name;
16365     }
16366     public void setName(String id) {
16367         this.name = id;
16368     }
16369     public String getClassInfo() {
16370         return classInfo;
16371     }
16372     public void setClassInfo(String classInfo) {
16373         this.classInfo = classInfo;
16374     }
16375     public int getType() {
16376         return type;
16377     }
16378     public void setType(int type) {
16379         this.type = type;
16380     }
16381 }
16382 package cn.edu.ustb.mt4ws.configuration;
16383 import java.util.Map;
16384 import cn.edu.ustb.mt4ws.tcg.TestSuite;
16385 public class Project {
```

```
16386     private Workspace workspace;
16387     private String projectName;
16388     private String wsdlUrl;
16389     protected Map<String, TestSuite> testSuites;
16390     public String getProjectName() {
16391         return projectName;
16392     }
16393     public String getWsdlUrl() {
16394         return wsdlUrl;
16395     }
16396     public void setWorkspace(Workspace workspace) {
16397         if (workspace == null){
16398             this.workspace = workspace;
16399         }else{
16400             this.workspace = workspace;
16401         }
16402     }
16403     public Workspace getWorkspace() {
16404         return workspace;
16405     }
16406     public Map<String, TestSuite> getTestSuiteMap() {
16407         return testSuites;
16408     }
16409     public void setProjectName(String projectName) {
16410         this.projectName = projectName;
16411     }
16412     public void setWSDLUri(String wsdlUri) {
16413         this.wsdlUrl = wsdlUri;
16414     }
16415     public void setTestSuites(Map<String, TestSuite> testsuites) {
16416         this.testSuites = testsuites;
16417     }
16418     public void print() {
16419         System.out.println("Project:" + projectName);
16420         System.out.println("WSDL URL=" + wsdlUrl);
16421         System.out.println("workspace=" + workspace.getPath());
16422     }
16423 }
16424 package cn.edu.ustb.mt4ws.configuration;
16425 import java.util.List;
16426 public class Configuration {
16427     private String operationName;
16428     private List<Integer> mrIds;
16429     private List<List<Integer>> testCases;
16430     public Configuration(){
16431     }
16432     public Configuration(String operationName, List<Integer> mrIds,
16433         List<List<Integer>> testCases) {
16434         this.operationName = operationName;
16435         this.mrIds = mrIds;
```

```
16436         this.testCases = testCases;
16437     }
16438     public void setOperationName(String operationName) {
16439         this.operationName = operationName;
16440     }
16441     public String getOperationName() {
16442         return operationName;
16443     }
16444     public void setMrIds(List<Integer> mrIds) {
16445         this.mrIds = mrIds;
16446     }
16447     public List<Integer> getMrIds() {
16448         return mrIds;
16449     }
16450     public void setTestCases(List<List<Integer>> testCases) {
16451         this.testCases = testCases;
16452     }
16453     public List<List<Integer>> getTestCases() {
16454         return testCases;
16455     }
16456 }
16457 package cn.edu.ustb.mt4ws.file;
16458 import java.io.File;
16459 import java.io.IOException;
16460 import javax.xml.transform.Source;
16461 import javax.xml.transform.stream.StreamSource;
16462 import javax.xml.validation.Schema;
16463 import javax.xml.validation.SchemaFactory;
16464 import javax.xml.validation.Validator;
16465 import org.xml.sax.SAXException;
16466 public class ValidatorXML {
16467     public ValidatorXML(){
16468     }
16469     public boolean validateXml(String xsdpath, String xmlpath)
16470         throws SAXException, IOException {
16471         SchemaFactory schemaFactory = SchemaFactory
16472             .newInstance("http://www.w3.org/2001/XMLSchema");
16473         File schemaFile = new File(xsdpath);
16474         Schema schema = schemaFactory.newSchema(schemaFile);
16475         Validator validator = schema.newValidator();
16476         Source source = new StreamSource(xmlpath);
16477         try {
16478             validator.validate(source);
16479         } catch (Exception ex) {
16480             ex.printStackTrace();
16481             return false;
16482         }
16483         return true;
16484     }
16485 }
```