Introduction

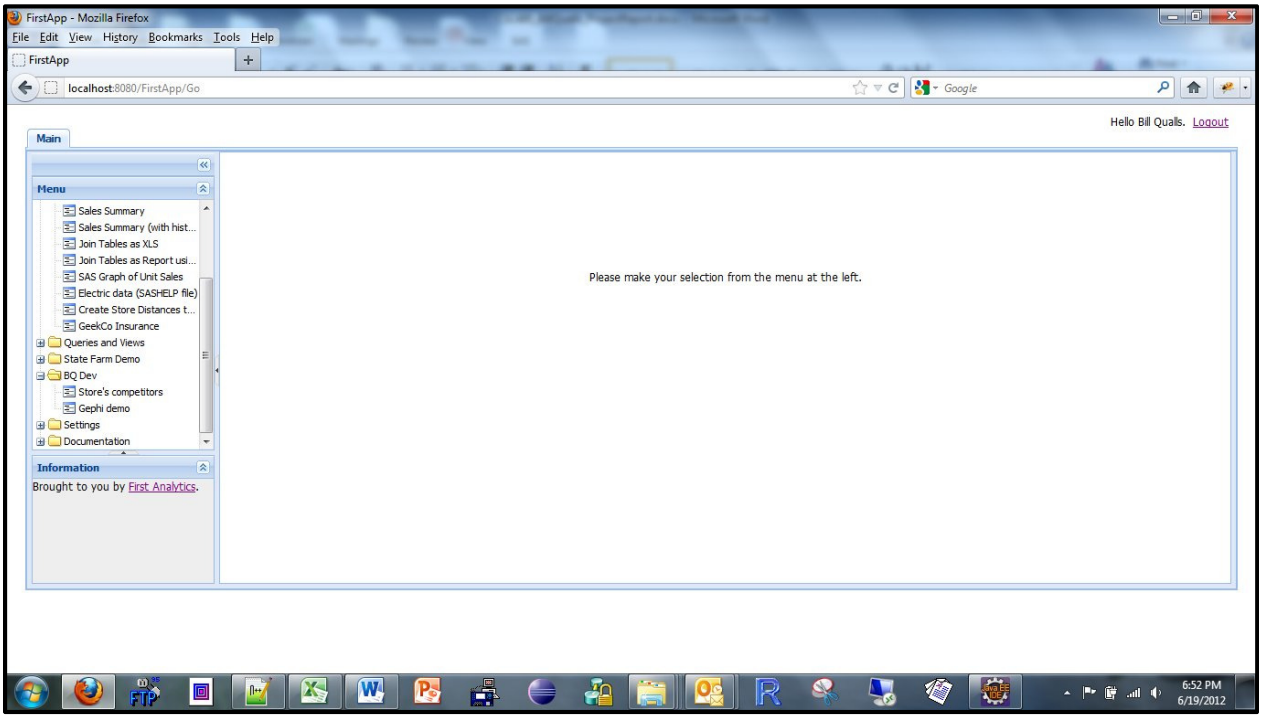
Gephi is an open source graph visualization and analysis tool. (See http://www.gephi.org.) My intent here is to provide a proof-of-concept on the usefulness of Gephi for graph (network) visualization. I have created a new option in FirstApp to demonstrate the visualization potential of Gephi. The input file is a comma-delimited file containing store distance data. Here is a subset of that file:

```
OPIS1,OPIS2,State,OPIS1IsPantry,OPIS2IsPantry,Distance
1889,1956,NC,N,N,102.76506312
1889,10365,NC,N,Y,23.789795246
1889,12134,NC,N,Y,46.839980487
1889,12899,NC,N,N,90.297555687
1889,16087,NC,N,N,114.21531605
```

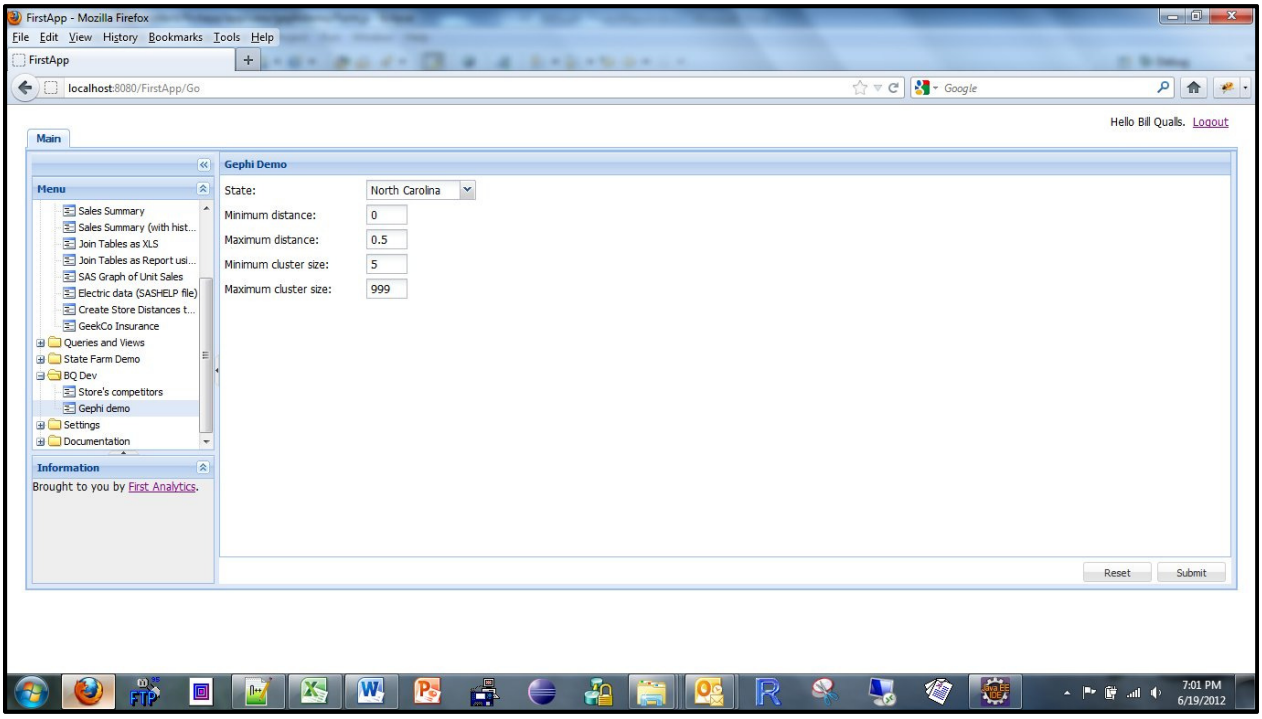
The program I have created prompts the user for the state, the minimum and maximum distance, and the minimum and maximum cluster size. It creates a graph of all stores meeting that criteria, and displays that graph as a PDF.

Screen shots

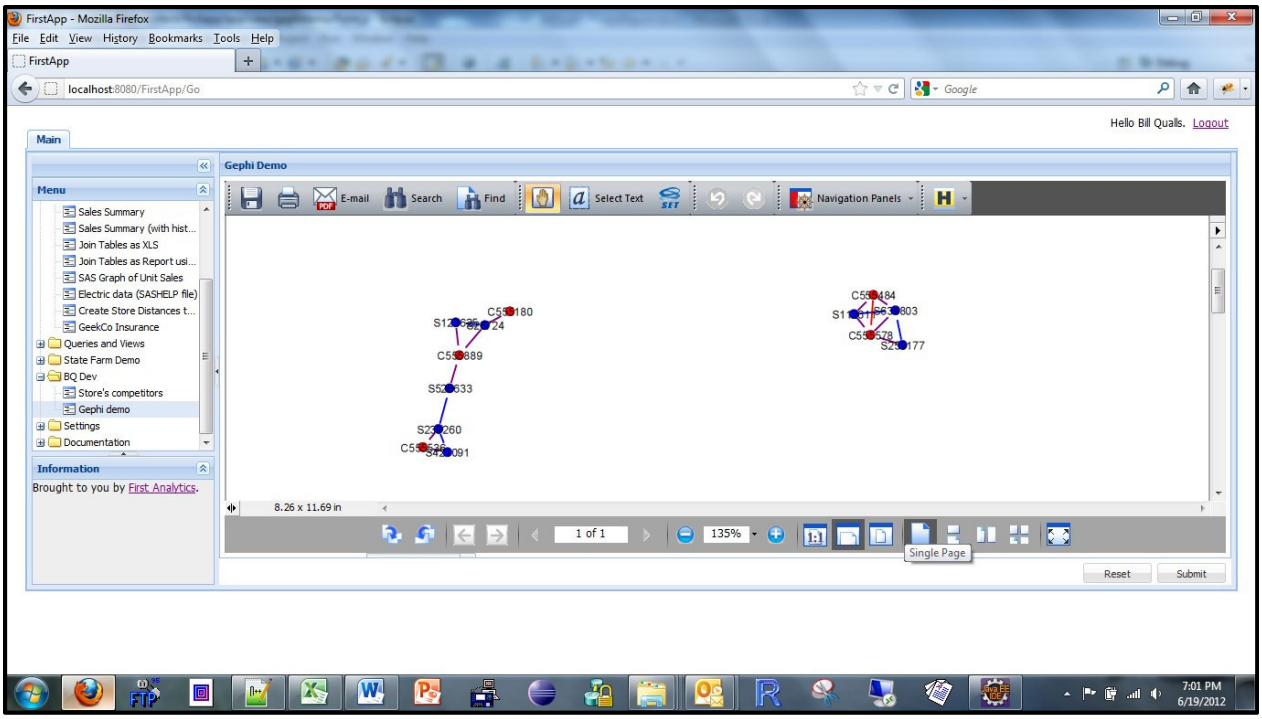
1) Choose Gephi Demo from the main menu.



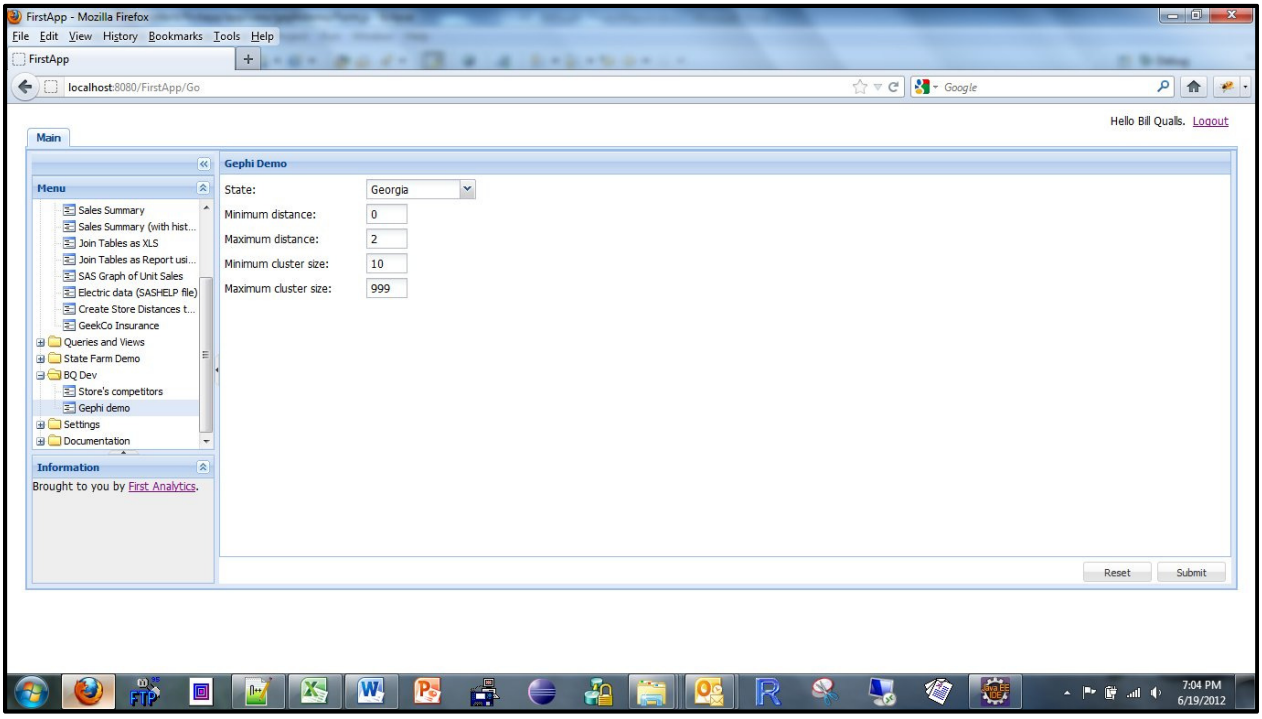
2) Specify criteria. Click Submit.



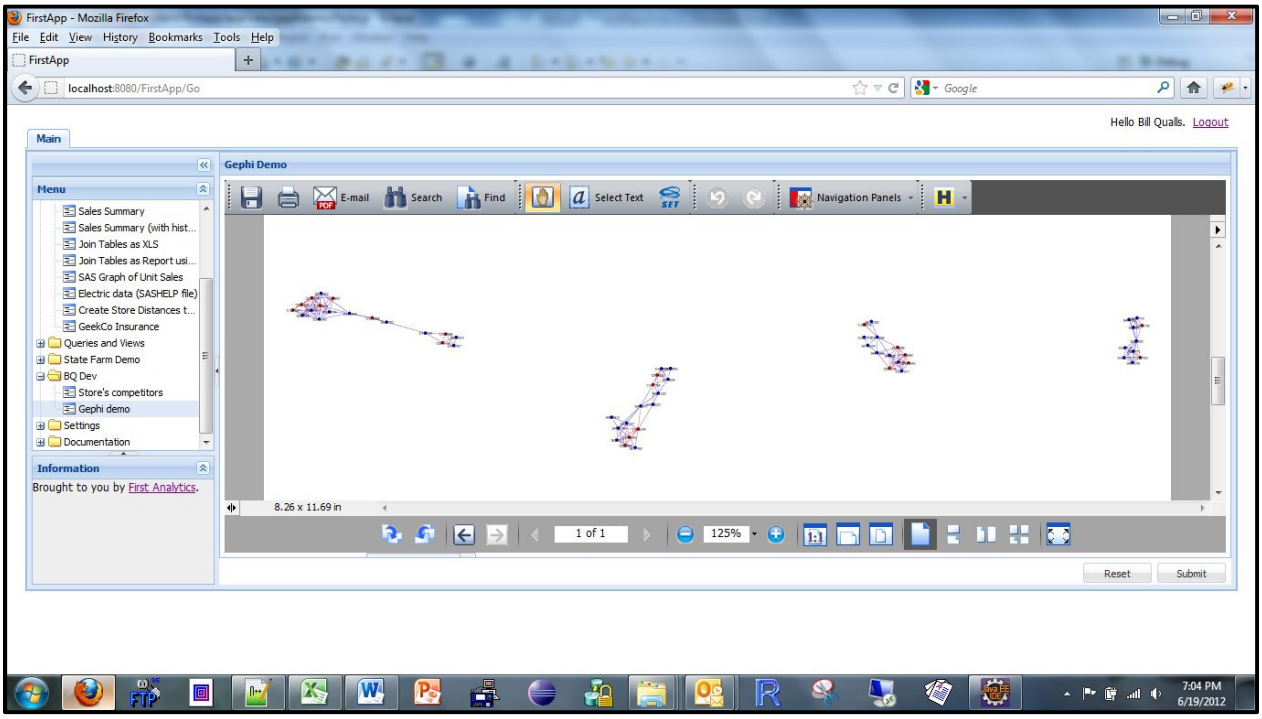
3) View resulting PDF. Note the node labels: client's own stores are blue and begin with "S" (as in "Self") followed by the store number, while competitor stores are red and begin with "C" (as in "Competitor") followed by the store number. Edges between client's own stores are blue, edges between competitors' stores are red, and edges between a client store and a competitor store are purple.



4) Press Reset. Specify other criteria. Click Submit.



5) View resulting PDF. Often, as in this case, it will be necessary to zoom in to read the node labels. The PDF format supports this zooming and the node labels can indeed by read easily. Unfortunately, the programmer has very little control over how the network will be displayed in the PDF.



Conclusion

I am not convinced that this particular application of Gephi provides any real value added for the present client. But I am convinced that such an opportunity will present itself in the future, with this client or another one. Gephi is another tool for our tool belt.

Source Code – Ext.js form

```
Ext.define('MyApp.view.gephidemo.Form' ,{
    extend: 'Ext.form.Panel',
    alias : 'widget.gephidemoform',

    // no store

    title: 'Gephi Demo',

    id: 'gephidemo.Form',

    bodyPadding: 5,
    autoScroll: false,  // false so the form is not seen when the PDF is shown
    layout: 'auto',

    defaults:
    {
        anchor: '100%'
    },

    items: [
    {
        xtype: 'combobox',
        fieldLabel: 'State',
        name: 'state',
        labelWidth: 150,
        displayField: 'display',
        valueField: 'state',
        store: Ext.create('Ext.data.Store', {
            fields: ['state', 'display'],
            data :
            [
                {"state":"AL", "display":"Alabama"},
                {"state":"FL", "display":"Florida"},
                {"state":"GA", "display":"Georgia"},
                {"state":"IN", "display":"Indiana"},
                {"state":"KS", "display":"Kansas"},
                {"state":"KY", "display":"Kentucky"},
                {"state":"LA", "display":"Louisiana"},
                {"state":"MS", "display":"Mississippi"},
                {"state":"MO", "display":"Missouri"},
                {"state":"NC", "display":"North Carolina"},
                {"state":"SC", "display":"South Carolina"},
                {"state":"TN", "display":"Tennessee"},
                {"state":"VA", "display":"Virginia"}
            ]
        }),
        mode: 'local',
        editable: false,
        width: 275,
        allowBlank: false
    },

    {
```

```
        xtype: 'numberfield',
        anchor: '100%',
        name: 'minDistance',
        fieldLabel: 'Minimum distance',
        minValue: 0, //prevents negative numbers
        labelWidth: 150,
        width: 200,  // includes labelWidth
        allowBlank: false,
        value: 0,
        // Remove spinner buttons, and arrow key and mouse wheel listeners
        hideTrigger: true, keyNavEnabled: false, mouseWheelEnabled: false
    },

    {
        xtype: 'numberfield',
        anchor: '100%',
        name: 'maxDistance',
        fieldLabel: 'Maximum distance',
        minValue: 0, //prevents negative numbers
        labelWidth: 150,
        width: 200,  // includes labelWidth
        allowBlank: false,
        value: .5,
        // Remove spinner buttons, and arrow key and mouse wheel listeners
        hideTrigger: true, keyNavEnabled: false, mouseWheelEnabled: false
    },

    {
        xtype: 'numberfield',
        anchor: '100%',
        name: 'minClusterSize',
        fieldLabel: 'Minimum cluster size',
        minValue: 1,
        labelWidth: 150,
        width: 200,  // includes labelWidth
        allowBlank: false,
        allowDecimals: false,
        value: 1,
        // Remove spinner buttons, and arrow key and mouse wheel listeners
        hideTrigger: true, keyNavEnabled: false, mouseWheelEnabled: false
    },

    {
        xtype: 'numberfield',
        anchor: '100%',
        name: 'maxClusterSize',
        fieldLabel: 'Maximum cluster size',
        minValue: 0, //prevents negative numbers
        labelWidth: 150,
        width: 200,  // includes labelWidth
        allowBlank: false,
        allowDecimals: false,
        value: 999,
        // Remove spinner buttons, and arrow key and mouse wheel listeners
        hideTrigger: true, keyNavEnabled: false, mouseWheelEnabled: false
```

```
    }
    ],

    // Reset and Submit buttons
    buttons: [
    {
        text: 'Reset',
        handler: function()
        {
            var thePanel = Ext.getCmp('id-gephidemo-panel');
            var pdfPanel = Ext.getCmp('id-gephidemo-pdf');
            if (pdfPanel)  // if defined
            {
                thePanel.remove(pdfPanel);
            }
            var form = this.up('form').getForm();
            form.reset();
        }
    },
    {
        text: 'Submit',
        formBind: true, //only enabled once the form is valid
        disabled: true,
        handler: function()
        {
            var form = this.up('form').getForm();
            if (form.isValid())
            {
                var box = Ext.MessageBox.wait('Create store network diagram...' +
                    'this can take a couple of minutes.', 'Please wait');
                var thePanel = Ext.getCmp('id-gephidemo-panel');
                var pdfPanel = Ext.getCmp('id-gephidemo-pdf');
                if (pdfPanel)  // if defined
                {
                    thePanel.remove(pdfPanel);
                }

                Ext.Ajax.request({
                    url: MyGlobalData.contextPath + '/GephiDemoServlet',
                    params : form.getValues(),
                    // because I used Ext.Ajax.request instead of form.submit
                    // which I had to do because I am returning HTML, not JSON.
                    timeout: 120000,
                    success: function(response)
                    {
                        var theText = response.responseText;
                        // cannot do thePanel.removeAll() because it would remove
                        // the form too!  instead, I used autoScroll: false so user
                        // cannot scroll down and see form.
                        thePanel.insert(0,
                        [
                            {
                                xtype: 'box',
                                id: 'id-gephidemo-pdf',
                                autoEl:
```

```
                                {
                                    tag: 'iframe',
                                    style: 'height: 100%; width: 100%; border: none',
                                    src: MyGlobalData.contextPath + theText
                                }
                            }
                        ]
                        );
                        box.hide();
                    },
                    failure: function(response)
                    {
                        box.hide();
                        Ext.Msg.alert('Failed...');
                    }
                });
            }  // end if
        }  // end handlet
    }  // end submit button
    ]  // end buttons
});  // Ext.define
```

Source Code – GephiDemoServlet.java

```
package servlets;

import java.awt.Color;
import java.io.BufferedReader;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.Map;
import java.util.Set;

import javax.servlet.ServletConfig;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import mybeans.Constants;
import mybeans.Pair;
import mybeans.Record;
import mybeans.Store;
import mybeans.SystemSettings;

import org.gephi.algorithms.shortestpath.DijkstraShortestPathAlgorithm;
import org.gephi.graph.api.Edge;
import org.gephi.graph.api.Graph;
import org.gephi.graph.api.GraphController;
import org.gephi.graph.api.GraphModel;
import org.gephi.graph.api.Node;
import org.gephi.graph.api.NodeIterator;
import org.gephi.graph.api.UndirectedGraph;
import org.gephi.io.exporter.api.ExportController;
import org.gephi.io.exporter.preview.PDFExporter;
import org.gephi.layout.plugin.forceAtlas2.ForceAtlas2;
import org.gephi.layout.plugin.forceAtlas2.ForceAtlas2Builder;
import org.gephi.preview.api.PreviewController;
import org.gephi.preview.api.PreviewModel;
import org.gephi.preview.api.PreviewProperty;
import org.gephi.preview.types.DependantOriginalColor;
import org.gephi.project.api.ProjectController;
import org.gephi.project.api.ProjectMetaData;
import org.openide.util.Lookup;

import security.Go;
import sharedmethods.SharedMethods;
```

```
import com.itextpdf.text.PageSize;

@WebServlet("/GephiDemoServlet")

public class GephiDemoServlet extends HttpServlet implements Constants
{
    private static final long serialVersionUID = 1L;

    // Files
    private static String DISTANCES_FILE_NAME = null;
    private String downloadDirectory = null;
    private String downloadPDFFileName = null;

    //Layout constants:
    private static final Double LAYOUT_SCALE = 200.0;
    private static final Boolean BARNES_HUTT_OPTIMIZE = true;
    private static final Boolean ADJUST_SIZES = true;
    private static final int LAYOUT_ITERATIONS = 10000;
    private static final Integer THREADS_COUNT = 3;

    // parameters
    String state;
    double minClusterSize;
    double maxClusterSize;
    double minDistance;
    double maxDistance;

    public GephiDemoServlet()
    {
        super();
    }

    public void init(ServletConfig config) throws ServletException
    {
        super.init(config);
        ServletContext context = config.getServletContext();
        String realPath = context.getRealPath(File.separator);
        System.out.println("realPath = " + realPath);
        DISTANCES_FILE_NAME = realPath + "data\\Distances.csv";
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        Go.securityCheck(request, response, this.getClass().getName());
        doPost(request, response);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        SharedMethods.log(request, "In " + this.getClass().getName() + ".doPost()");

        String userId = SharedMethods.getUserIdFromSession(request);
        int companyId = SharedMethods.getCompanyIdFromSession(request);
```

```
        state = request.getParameter("state");
        minClusterSize = Double.parseDouble(request.getParameter("minClusterSize"));
        maxClusterSize = Double.parseDouble(request.getParameter("maxClusterSize"));
        minDistance = Double.parseDouble(request.getParameter("minDistance"));
        maxDistance = Double.parseDouble(request.getParameter("maxDistance"));

        downloadDirectory = SystemSettings.get_LOC_OF_FILES_FOR_DOWNLOAD();
        downloadPDFFileName = SharedMethods.generateDownloadFileName(companyId,
            userId, "pdf");

        script();

        String returnMessage = DOWNLOAD_FOLDER + downloadPDFFileName;
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.print(returnMessage);
        out.flush();
    }

    private void setupController (PreviewController controller)
    {
        PreviewModel previewModel = controller.getModel();
        previewModel.getProperties().putValue(PreviewProperty.SHOW_NODE_LABELS,
            Boolean.TRUE);
        previewModel.getProperties().putValue(PreviewProperty.NODE_LABEL_COLOR,
            new DependantOriginalColor(Color.BLACK));
        previewModel.getProperties().putValue(PreviewProperty.EDGE_CURVED,
            Boolean.FALSE);
        previewModel.getProperties().putValue(PreviewProperty.EDGE_OPACITY, 100);
        previewModel.getProperties().putValue(PreviewProperty.EDGE_RADIUS, 2f);
        previewModel.getProperties().putValue(PreviewProperty.EDGE_THICKNESS, 5f);
        previewModel.getProperties().putValue(PreviewProperty.ARROW_SIZE, 3.0f);
        previewModel.getProperties().putValue(PreviewProperty.BACKGROUND_COLOR,
            Color.WHITE);
        controller.refreshPreview();
    }

    public static Node getNode(Graph graph, Store store)
    {
        Node answer = null;
        NodeIterator iter = graph.getNodes().iterator();
        while(iter.hasNext())
        {
            Node n = iter.next();
            if (n.getNodeData().getLabel().equals(store.toString()))
            {
                answer = n;
                break;
            }
        }
        graph.readUnlockAll();
        return answer;
    }
```

```
public void script()
{
    GraphModel m_graphModel = null;
    UndirectedGraph m_graph = null;

    //Init a project - and therefore a workspace
    ProjectController pc = Lookup.getDefault().lookup(ProjectController.class);
    pc.newProject();

    ProjectMetaData metaData =
        pc.getCurrentProject().getLookup().lookup(ProjectMetaData.class);
    m_graphModel = Lookup.getDefault().lookup(GraphController.class).getModel();
    m_graph = m_graphModel.getUndirectedGraph();
    fillGraph(m_graph);

    //Preview configuration
    PreviewController previewController =
        Lookup.getDefault().lookup(PreviewController.class);

    setupController (previewController);

    ExportController pdfExportController =
        Lookup.getDefault().lookup(ExportController.class);
    try
    {
        pdfExportController.exportFile(new File(downloadDirectory +
            downloadPDFFileName));
    }
    catch(Exception ex)
    {
        System.out.println("Error creating pdfExportController");
        ex.printStackTrace();
        return;
    }

    //PDF Exporter config and export to Byte array
    PDFExporter pdfExporter = (PDFExporter) pdfExportController.
        getExporter("pdf");
    pdfExporter.setPageSize(PageSize.A0);
    ByteArrayOutputStream baos = new ByteArrayOutputStream();
    pdfExportController.exportStream(baos, pdfExporter);
}

public void fillGraph(UndirectedGraph graph)
{
    GraphModel model = graph.getGraphModel();
    ArrayList<Store> stores = new ArrayList<Store>();
    ArrayList<Pair> pairs = new ArrayList<Pair>();

    try
    {
        String csv = null;
        File infile = new File(DISTANCES_FILE_NAME);
        FileInputStream fis = new FileInputStream(infile);
        InputStreamReader isr = new InputStreamReader(fis);
```

```
        BufferedReader br = new BufferedReader(isr);
        br.readLine();  // skip header record
        while ((csv = br.readLine()) != null)
        {
            Record record = new Record(csv);
            if ((record.getDistance() >= minDistance)
                && (record.getDistance() <= maxDistance)
                && (record.getState().equalsIgnoreCase(state)))
            {
                Store.addStoreToList(stores, new Store(record.getOpis1(),
                    record.getOpis1IsSelf()));
                Store.addStoreToList(stores, new Store(record.getOpis2(),
                    record.getOpis2IsSelf()));
                pairs.add(new Pair(stores, record.getOpis1(), record.getOpis2(),
                    record.getDistance()));
            }
        }
    }
    catch (IOException ioe)
    {
        System.out.println("Error reading file " + DISTANCES_FILE_NAME);
        System.out.println(ioe);
        ioe.printStackTrace();
        System.exit(1);
    }

    // create node for each store
    for (int i = 0; i < stores.size(); i++)
    {
        Store s = stores.get(i);
        Node n = model.factory().newNode(s.toString());
        String label = s.toString();
        n.getNodeData().setLabel(label);
        n.getNodeData().setSize(16);
        // self is blue, competitor is red
        if (s.isCompetitor())
        {
            n.getNodeData().setColor(1, 0, 0);
        }
        else
        {
            n.getNodeData().setColor(0, 0, 1);
        }
        graph.addNode(n);
    }

    // create edge for each pair
    for (int i = 0; i < pairs.size(); i++)
    {
        Pair p = pairs.get(i);
        Node fromNode = getNode(model.getGraph(), p.getFromStore());
        Node toNode = getNode(model.getGraph(), p.getToStore());
        Edge e = model.factory().newEdge(fromNode, toNode);
        graph.addEdge(e);
```

```
        }

        // Show nodes and component size
        Node [] nodes = new Node[stores.size()];
        int [] componentSize = new int[stores.size()];
        int j = 0;
        for (Node n: graph.getNodes())
        {
            nodes[j] = n;
            componentSize[j] = findComponentSize(n, graph);
            j++;
        }

        int nodesDeletedDueToClusterSize = 0;
        for (int i = 0; i < nodes.length; i++)
        {
            if (componentSize[i] < minClusterSize
                || componentSize[i] > maxClusterSize)
            {
                graph.removeNode(nodes[i]);
                nodesDeletedDueToClusterSize++;
            }
        }
        System.out.println("Deleted " + nodesDeletedDueToClusterSize +
            " nodes due to cluster size.");

        // Source: https://forum.gephi.org/viewtopic.php?f=26&t=1590&start=0
        //Run Force Atlas 2 layout - The layout always takes the current visible view
        ForceAtlas2 layout = new ForceAtlas2Builder().buildLayout();
        layout.setGraphModel(model);
        layout.initAlgo();
        layout.resetPropertiesValues();
        layout.setAdjustSizes(ADJUST_SIZES);
        layout.setBarnesHutOptimize(BARNES_HUTT_OPTIMIZE);
        layout.setScalingRatio(LAYOUT_SCALE);
        layout.setThreadsCount(THREADS_COUNT);

        for (int i = 0; i < LAYOUT_ITERATIONS && layout.canAlgo(); i++)
        {
            layout.goAlgo();
        }
        layout.endAlgo();
    }

    // getNodeSet taken from CSC495 lab 1
    public static Set<Node> getNodeSet (UndirectedGraph graph)
    {
        Set<Node> ns = new HashSet<Node>();
        for (Node n: graph.getNodes())
        {
            ns.add(n);
        }
        return ns;
    }
```

```
    public int findComponentSize(Node node, UndirectedGraph graph)
    {
        int componentSize = 0;
        DijkstraShortestPathAlgorithm alg =
            new DijkstraShortestPathAlgorithm(graph, node);
        alg.compute();
        Map<Node, Double> distMap = alg.getDistances();
        for (Node reachable: distMap.keySet())
        {
            if (distMap.get(reachable) != Double.POSITIVE_INFINITY)
            {
                componentSize++;
            }
        }
       return componentSize;
    }
}
```

Source Code – Record.java

```
package mybeans;

import java.util.StringTokenizer;

public class Record
{
    private int opis1;
    private int opis2;
    private String state;
    private boolean opis1IsSelf;
    private boolean opis2IsSelf;
    private double distance;

    public Record(String csv)
    {
        try
        {
            StringTokenizer tokens = new StringTokenizer(csv, ",");
            int opis1 = Integer.parseInt(tokens.nextToken());
            int opis2 = Integer.parseInt(tokens.nextToken());
            String state = tokens.nextToken();
            boolean opis1IsSelf = "Y".equals(tokens.nextToken());
            boolean opis2IsSelf = "Y".equals(tokens.nextToken());
            double distance = 999;  // for missing distances
            if (tokens.hasMoreTokens())
            {
                distance = Double.parseDouble(tokens.nextToken());
            }
            setAll(opis1, opis2, state, opis1IsSelf, opis2IsSelf, distance);
        }
        catch (Exception e)
        {
            System.out.println("Parse error: " + csv);
        }
    }

    public Record(int opis1, int opis2, String state, boolean opis1IsSelf,
        boolean opis2IsSelf, double distance)
    {
        setAll(opis1, opis2, state, opis1IsSelf, opis2IsSelf, distance);
    }

    private void setAll(int opis1, int opis2, String state, boolean opis1IsSelf,
        boolean opis2IsSelf, double distance)
    {
        // method setAll is used by constructors.

        // opis1 will always be less than opis2.  Cuts list size in half.
        if (opis1 < opis2)
        {
            setOpis1(opis1);
            setOpis1IsSelf(opis1IsSelf);
            setOpis2(opis2);
```

```
            setOpis2IsSelf(opis2IsSelf);
        }
        else
        {
            setOpis1(opis1);
            setOpis1IsSelf(opis1IsSelf);
            setOpis2(opis2);
            setOpis2IsSelf(opis2IsSelf);
        }
        setState(state);
        setDistance(distance);
    }

    public int getOpis1()
    {
        return opis1;
    }

    public void setOpis1(int opis1)
    {
        this.opis1 = opis1;
    }

    public int getOpis2()
    {
        return opis2;
    }

    public void setOpis2(int opis2)
    {
        this.opis2 = opis2;
    }

    public String getState()
    {
        return state;
    }

    public void setState(String state)
    {
        this.state = state;
    }

    public boolean getOpis1IsSelf()
    {
        return opis1IsSelf;
    }

    public void setOpis1IsSelf(boolean opis1IsSelf)
    {
        this.opis1IsSelf = opis1IsSelf;
    }

    public boolean getOpis2IsSelf()
    {
```

```
        return opis2IsSelf;
    }

    public void setOpis2IsSelf(boolean opis2IsSelf)
    {
        this.opis2IsSelf = opis2IsSelf;
    }

    public double getDistance()
    {
        return distance;
    }

    public void setDistance(double distance)
    {
        this.distance = distance;
    }

}
```

Source Code – Store.java

```
package mybeans;

import java.util.List;

public class Store
{
    private int storeId;
    private boolean competitor;

    public Store(int storeId, boolean competitor)
    {
        this.storeId = storeId;
        this.competitor = competitor;
    }

    public int getStoreId()
    {
        return this.storeId;
    }

    public boolean isCompetitor()
    {
        return this.competitor;
    }

    public String toString()
    {
        // (S)elf or (C)ompetitor
        return (isCompetitor()? "C": "S") + getStoreId();
    }

    public static boolean isStoreIDInList(List<Store> list, int storeId)
    {
        for (int i = 0; i < list.size(); i++)
        {
            Store s = list.get(i);
            if (storeId == s.getStoreId())
            {
                return true;
            }
        }
        return false;
    }

    public static boolean addStoreToList(List<Store> list, Store store)
    {
        boolean answer;
        if (!isStoreIDInList(list, store.getStoreId()))
        {
            list.add(store);
            answer = true;
        }
        else
```

```
        {
            answer = false;
        }
        return answer;
    }

    public static Store getStoreFromList(List<Store> list, int storeId)
    {
        for (int i = 0; i < list.size(); i++)
        {
            Store s = list.get(i);
            if (storeId == s.getStoreId())
            {
                return s;
            }
        }
        return null;
    }

}
```

Source Code – Pair.java

```
package mybeans;

import java.text.DecimalFormat;
import java.util.List;

public class Pair
{
    public static final DecimalFormat DISTANCE_FORMAT = new DecimalFormat("0.000");

    private Store fromStore;
    private Store toStore;
    private double distance;

    // two different constructors...
    public Pair(List<Store> list, int fromStoreId, int toStoreId, double distance)
    {
        this.fromStore = Store.getStoreFromList(list, fromStoreId);
        this.toStore = Store.getStoreFromList(list, toStoreId);
        this.distance = distance;
    }

    public Pair(Store fromStore, Store toStore, double distance)
    {
        this.fromStore = fromStore;
        this.toStore = toStore;
        this.distance = distance;
    }

    public Store getFromStore()
    {
        return this.fromStore;
    }

    public Store getToStore()
    {
        return this.toStore;
    }

    public double getDistance()
    {
        return this.distance;
    }

    public String toString()
    {
        return "(" + fromStore.getStoreId() + "," +
            toStore.getStoreId() + ")=" +
            DISTANCE_FORMAT.format(getDistance());
    }

    public static boolean isPairInList(List<Pair> list, int store1, int store2)
    {
        for (int i = 0; i < list.size(); i++)
```

```
        {
            Pair pair = list.get(i);
            if ((store1 == pair.getFromStore().getStoreId()
                   && store2 == pair.getToStore().getStoreId())
                || (store2 == pair.getFromStore().getStoreId()
                   && store1 == pair.getToStore().getStoreId()))
            {
                return true;
            }
        }
        return false;
    }

    public static boolean addPairToList(List<Pair> list, Pair pair)
    {
        boolean answer;
        if (!isPairInList(list, pair.getFromStore().getStoreId(),
               pair.getToStore().getStoreId()))
        {
            list.add(pair);
            answer = true;
        }
        else
        {
            answer = false;
        }
        return answer;
    }

}
```