

Οπτικοποίηση με Prefuse

Δομή / Βασικά Χαρακτηριστικά / Παράδειγμα

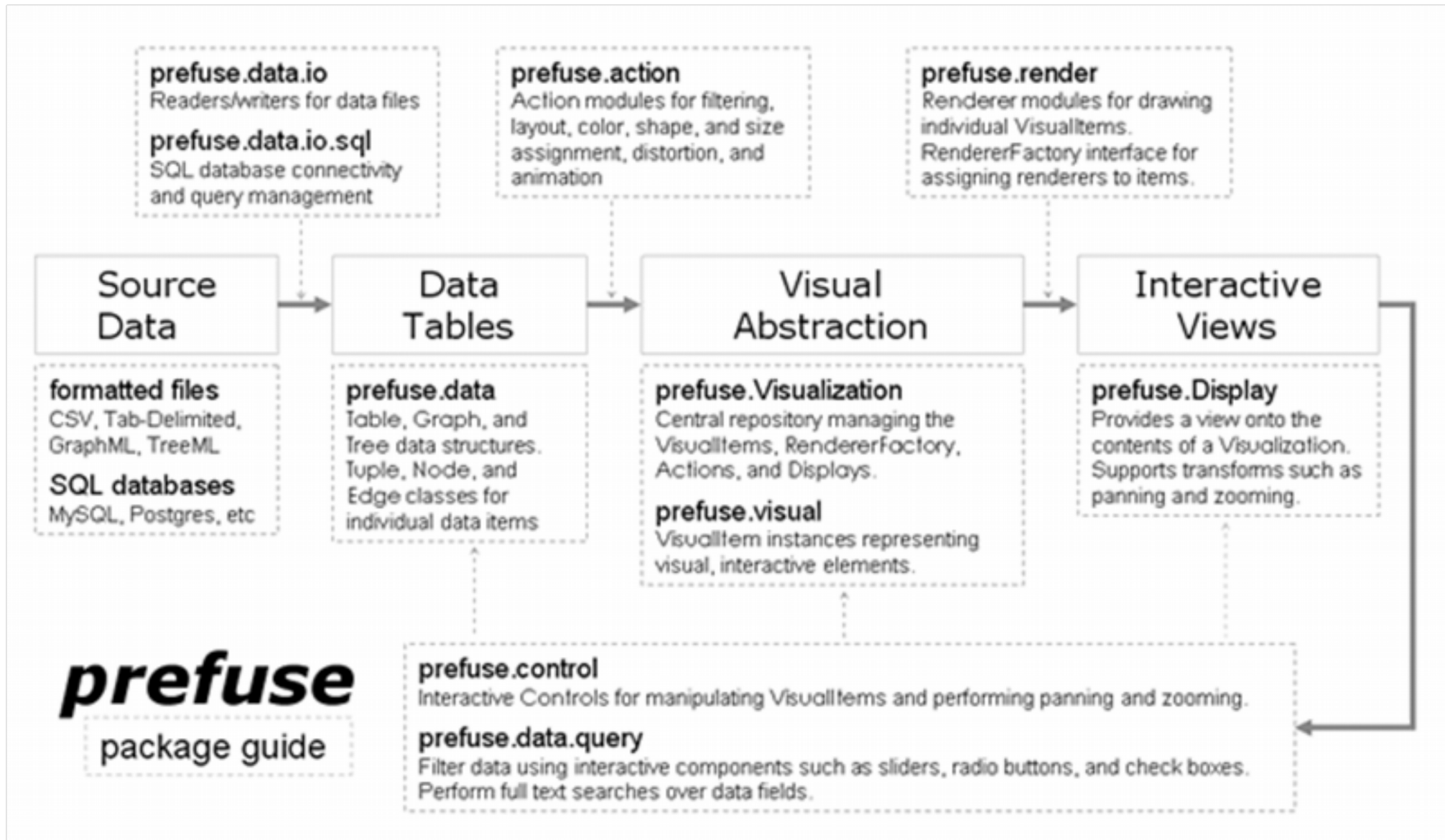
4 Βασικά Βήματα

- 1) Εισαγωγή των δεδομένων σε εσωτερικές δομές Prefuse
- 2) Καθορισμός του visual Abstraction
- 3) Δημιουργία View
- 4) Προσθήκη των δυνατοτήτων για διαχείριση

MVC

Prefuse και το μοντέλο Model View Controller:

- **Model**: internal Prefuse data structures
- **View**: Visual Abstraction
- **Controller**: δυνατότητες user Interaction



1) Εισαγωγή Δεδομένων

Το πρόγραμμα κάνει import:

- `import prefuse.data.*;`
- `import prefuse.data.io.*;`

Τα δεδομένα, ανάλογα με την πηγή από την οποία προέρχονται διαβάζονται σε δομή πίνακα, γράφου ή δένδρου.

- Όταν γεμίζουμε πίνακα τα δεδομένα εισάγονται σε αντικείμενα της κλάσης ***Tuple***. Οι γράφοι και τα δένδρα χρησιμοποιούν τις κλάσεις ***Node*** και ***Edge***.

SQL

Το Prefuse μπορεί να φορτώνει πίνακες από σχεσιακές βάσεις δεδομένων μέσω της:

- [prefuse.data.io.sql](#)

Παράδειγμα Εισαγωγής Δεδομένων

```
// 1. Load the data
```

```
Graph graph = null;  
/* graph will contain the core data */  
try {  
    graph = new GraphMLReader().readGraph("data/socialnet.xml");  
/* load the data from an XML file */  
} catch (DataIOException e) {  
    e.printStackTrace();  
    System.err.println("Error loading graph. Exiting...");  
    System.exit(1);  
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- An excerpt of an egocentric social network -->
<graphml xmlns="http://graphml.graphdrawing.org/xmlns">
<graph edgedefault="undirected">

<!-- data schema -->
<key id="name" for="node" attr.name="name" attr.type="string"/>
<key id="gender" for="node" attr.name="gender" attr.type="string"/>

<!-- nodes -->
<node id="1">
  <data key="name">Jeff</data>
  <data key="gender">M</data>
</node>
<node id="2">
  <data key="name">Ed</data>
  <data key="gender">M</data>
</node>
<node id="3">
  <data key="name">Christiaan</data>
  <data key="gender">M</data>
</node>
<node id="4">
  <data key="name">Emily</data>
  <data key="gender">F</data>
</node>
```

```
<!-- edges -->  
<edge source="1" target="2"></edge>  
<edge source="1" target="3"></edge>  
<edge source="1" target="4"></edge>  
<edge source="1" target="5"></edge>  
<edge source="1" target="6"></edge>  
<edge source="1" target="7"></edge>  
<edge source="1" target="8"></edge>  
<edge source="1" target="9"></edge>  
<edge source="1" target="10"></edge>  
<edge source="1" target="11"></edge>  
<edge source="1" target="12"></edge>  
<edge source="1" target="13"></edge>  
<edge source="1" target="14"></edge>  
<edge source="1" target="15"></edge>  
<edge source="1" target="16"></edge>  
<edge source="1" target="17"></edge>  
<edge source="1" target="18"></edge>
```

2) Visual Abstraction

- Κλάση **Visualization**: Πρόκειται για την σημαντικότερη κλάση στο Prefuse, επειδή είναι αυτή η οποία κάνει την οπτικοποίηση.

Π.χ.

```
Table t = new CSVTableReader().readTable("data/population.csv");
```

```
Visualization vis = new Visualization();
```

```
VisualTable vt = vis.addTable("myTable", t);
```

- Το vt είναι μία αναφορά στο visual abstraction του πίνακα t, στην οποία συνυπάρχουν τα δεδομένα του t, μαζί με δεδομένα απεικόνισης όπως x,y συντεταγμένες, χρώμα, μέγεθος και τιμές γραμματοσειρών

```
// 3. setup the renderers and the render factory

// labels for name
LabelRenderer nameLabel = new LabelRenderer("name");
nameLabel.setRoundedCorner(8, 8);
/* nameLabel describes how to draw the data elements labeled as "name" */

// create the render factory
vis.setRendererFactory(new DefaultRendererFactory(nameLabel));
```

```
// 4. process the actions

// colour palette for nominal data type
int[] palette = new int[]{ColorLib.rgb(255, 180, 180), ColorLib.rgb(190, 190, 255)};
/* ColorLib.rgb converts the colour values to integers */

// map data to colours in the palette
DataColorAction fill = new DataColorAction("socialnet.nodes", "gender",
Constants.NOMINAL, VisualItem.FILLCOLOR, palette);
/* fill describes what colour to draw the graph based on a portion of the data */

// node text
ColorAction text = new ColorAction("socialnet.nodes", VisualItem.TEXTCOLOR,
ColorLib.gray(0));
/* text describes what colour to draw the text */

// edge
ColorAction edges = new ColorAction("socialnet.edges", VisualItem.STROKECOLOR,
ColorLib.gray(200));
/* edge describes what colour to draw the edges */
```

```
// combine the colour assignments into an action list
ActionList colour = new ActionList();
colour.add(fill);
colour.add(text);
colour.add(edges);
vis.putAction("colour", colour);
/* add the colour actions to the visualization */

// create a separate action list for the layout
ActionList layout = new ActionList(Activity.INFINITY);
layout.add(new ForceDirectedLayout("socialnet"));
/* use a force-directed graph layout with default parameters */

layout.add(new RepaintAction());
/* repaint after each movement of the graph nodes */

vis.putAction("layout", layout);
/* add the layout actions to the visualization */
```

```
// 5. add interactive controls for visualization
```

```
Display display = new Display(vis);  
display.setSize(700, 700);  
display.pan(350, 350);    // pan to the middle  
display.addControlListener(new DragControl());  
/* allow items to be dragged around */
```

```
display.addControlListener(new PanControl());  
/* allow the display to be panned (moved left/right, up/down) (left-drag)*/
```

```
display.addControlListener(new ZoomControl());  
/* allow the display to be zoomed (right-drag) */
```

```
// 6. launch the visualizer in a JFrame
```

```
JFrame frame = new JFrame("prefuse tutorial: socialnet");  
/* frame is the main window */
```

```
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
frame.add(display);  
/* add the display (which holds the visualization) to the window */
```

```
frame.pack();  
frame.setVisible(true);
```

```
/* start the visualization working */  
vis.run("colour");  
vis.run("layout");
```