ImageJ Legacy

center for systems biology dresden

dais

de.NBI
GERMAN NETWORK FOR BIOINFORMATICS INFRASTRUCTURE

Matthias Arzt
maarzt
arzt@mpi-cbg.de

Session Overview

- Why legacy? (5 Slides)
- How to distinguish IJ1 / IJ2? (2 Slides)
- IJ1 and IJ2 Versions of: (4 Slides)
  - Images
  - Rois
  - (Services)
  - Tables
- 4 Exercises

Why ImageJ2 ?

- Modern Software Architecture
- UI Independent / Headless
- Extensible
- Imglib 2

Advantage of ImgLib2

IJ1

```
// normalize all pixels
for (int t = 1; t <= output.getNFrames(); t++) {
    input.setT(t);
    output.setT(t);
    for (int c = 1; c <= output.getNChannels(); c++) {
        input.setC(c);
        output.setC(c);
        for (int z = 1; z <= output.getNSlices(); z++) {
            input.setZ(z);
            output.setZ(z);
            ImageProcessor inputProcessor = input.getProcessor();
            ImageProcessor outputProcessor = output.getProcessor();
            for (int x = 0; x < output.getWidth(); x++) {
                for (int y = 0; y < output.getWidth(); y++) {
                    float value = inputProcessor.getf(x, y);
                    float normalisedValue = (value - minPixelValue)
                         / (maxPixelValue - minPixelValue);

                    outputProcessor.setf(x, y, normalisedValue);
                }
            }
        }
    }
}
```

ImgLib2

```
// normalize all pixels
Cursor<T> inputCursor = Views.flatIterable(input).cursor();
Cursor<FloatType> outputCursor = output.cursor();
while (inputCursor.hasNext() && outputCursor.hasNext()) {
    float value = inputCursor.next().getRealFloat();
    float normalisedValue = (value - minPixelValue) /
                (maxPixelValue - minPixelValue);

    outputCursor.next().set(normalisedValue);
}
```

Why ImageJ1

- UI
- Mature Software
- Useful / Widely Used Tool

- Limited Extensibility

FIJI is IJ1 + IJ2

Thats crazy! How does it work?

FIJI is IJ1 + IJ2

imagej-legacy

```
<dependency>
	<groupId>net.imagej</groupId>
	<artifactId>imagej-legacy</artifactId>
</dependency>
```

Injects code into IJ2:

- Main Menu
- PlugIn Execution IJ.run(...)
- Image Opening
- ...

Extends IJ2

- LegacyService
- LegacyDisplayService
- Converters:
  - Images
  - Table
  - Rois
  - ...

Why is legacy still important?

Because of:

- Macro Recorder
- Save + Open Images
  (Sometimes faster + weird)
- Missing Functionality:
  - Rois (done)
  - Tables (done)
  - Progress Bar (done)

How to distinguish?

IJ1

```
import ij.…
IJ.…

new ij.ImageJ()

PlugInFilter

ImagePlus
ImageStack
ImageProcessor
ImageWindow
Roi
Overlay
```

IJ2

```
import net.imagej.…
import net.imglib2.…
import com.scijava.…

new net.imagej.ImageJ()
Context, …Service

Command
@Plugin, @Paramter

Img, Dataset
RandomAccessibleInterval
FloatType, UnsigneByteType…
```

How to distinguish?

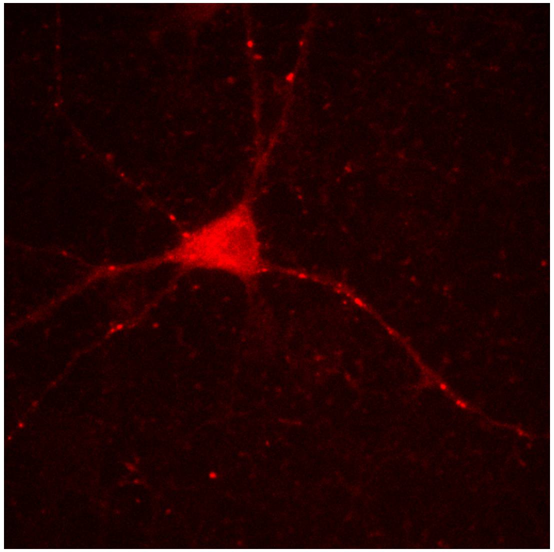
IJ1 vs. IJ2 Plugin

```
public class ImageNormalizerPlugin
    implements PlugInFilter
{


    @Override
    public int setup(String s, ImagePlus imagePlus)
    {
        return DOES_8G + DOES_16 + DOES_32;
    }


    @Override
    public void run(ImageProcessor imageProcessor)
    {
        ImagePlus input = IJ.getImage();
        // process …
        ImagePlus output = …;
        output.show();
    }
}
```

```
Config File:
* plugins.config
```

```
@Plugin(type = Command.class,
        menuPath = "Plugins>Filtering>Normalisation")
public class ImageNormalizerPlugin
        implements Command
{

    @Parameter
    Dataset input;

    @Parameter( type = ItemIO.OUTPUT )
    ImageJ output;

    @Override
    public void run()
    {
        // process ...
        output = ...;
    }

}
```

Images



Images

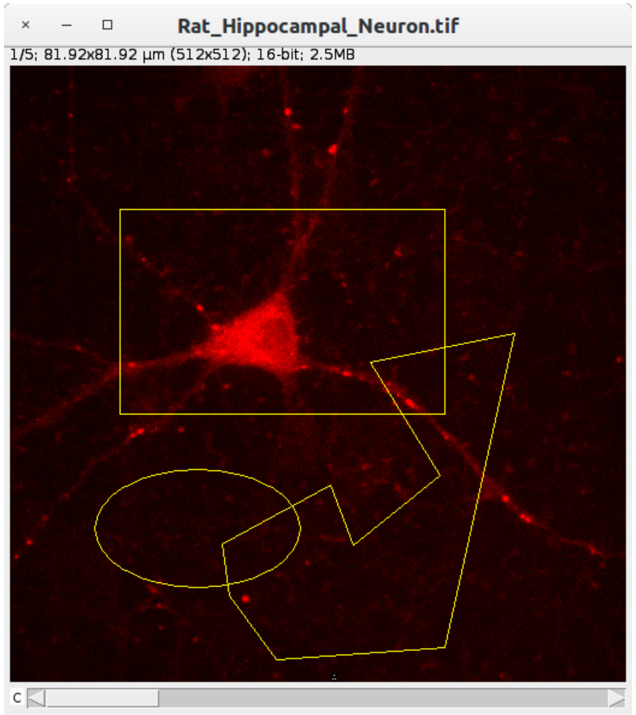
IJ1	IJ2
ImagePlus	Dataset
ImageStack ImageProcessor	Img & RandomAccessibleInterval

Conversion:

- ConvertService (not recommended)

```
→ Img<?> img = ImageJFunctions.wrap(imagePlus);
→ Img<? Extends RealType<?>> img = ImageJFunctions.wrap(imagePlus);
← ImagePlus imagePlus = ImageJFunctions.wrap(img, “title”)
```

Rois
Regions Of Interest



```
1/5; 81.92x81.92 µm (512x512); 16-bit; 2.5MB
```

Rois

IJ1	IJ2
Roi	RealRoi
Overlay (collection of Roi)	ROITree (tree of ???)

Conversion:

- ConvertService (recommended)

```
→ RealRoi roi2 = convertService.convert(roi1, RealRoi.class);
← Roi roi1 = convertService.conver(roi2, Roi.class);
```

- Note: `imagePlus.getRoi(), imagePlus.getOverlay()`

Services

IJ1	IJ2
IJ.openImage(…)	DatasetIOService
IJ.save(…)	IOService
IJ.log(…)	LogService / Logger
IJ.show(…)	UIService
IJ.run(...)	CommansService, OpService
IJ.showProgress()	StatusService

Gateway

```
ImageJ imageJ = new ImageJ();
imageJ.ui()
```

In a Command

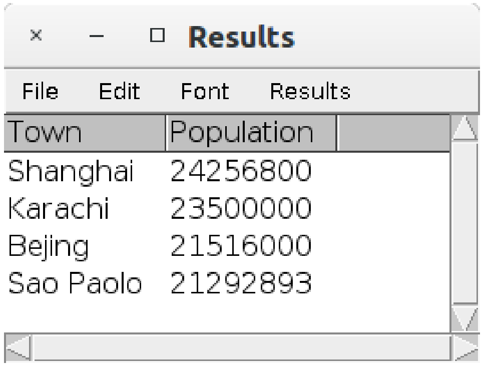
```
@Parameter
UIService uiService;
```

From the Context

```
Context context = …;
context.service(UIService.class)
```

Tables

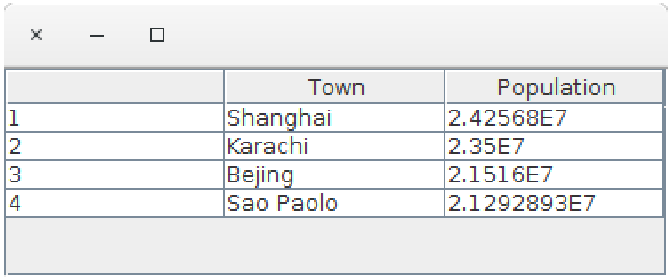
IJ1



Town	Population
Shanghai	24256800
Karachi	23500000
Bejing	21516000
Sao Paolo	21292893

ResultsTable

IJ2



	Town	Population
1	Shanghai	2.42568E7
2	Karachi	2.35E7
3	Bejing	2.1516E7
4	Sao Paolo	2.1292893E7

GenericTable

Conversion:

- Cheetsheet
- ConvertService (recommended)

```
→ RealRoi roi2 = convertService.convert(roi1, RealRoi.class);
```

Links

- Exercises:
  - https://github.com/maarzt/imagej-legacy-course
- Cheat sheet:
  - https://github.com/mpicbg-scicomp/ij2course-images/blob/master/slides/ij_legacy_cheetsheet.pdf
- ImageJ Legacy Course focused on:
  Updating Image Processing Pipeline IJ1 → IJ2
  - https://github.com/mpicbg-scicomp/ij2course-images
  - https://github.com/mpicbg-scicomp/ij2course-regions
  - https://github.com/mpicbg-scicomp/ij2course-tables