Input and Output System - Import

Molecule import is the operation when sources of data, i.e. structures defined in various formats are converted to Molecule objects so that ChemAxon applications can operate with them.

Sources of data

When importing structures with ChemAxon tools, we will refer to different sources the data comes from:

- Structure file where location is given with absolute or relative path
- Structure file where location is given with URL
- Molecule source text (e.g. pre-read content of a structure file) in various formats
- Molecule object from another application or database

Basic import using the API

The most frequently used API for molecule import is defined in chemaxon.formats.MolImporter class. `MolImporter` has lots of utility functions.

Importing from String

The simplest way of importing one molecule where the molecule source is available as String is using the static method of `MolImporter` class.

```
try {
        Molecule mol = MolImporter.importMol("CCC(N)c1cc(Cl)cc(C(N)CC)c1Br");
    // do something with the molecule
} catch (MolFormatException e) {
    // handle the exception
}
```

Working example of reading a molecule from string

```
/*
 * Copyright (c) 1998-2014 ChemAxon Ltd. All Rights Reserved.
 */

import chemaxon.formats.MolImporter;
import chemaxon.formats.MolFormatException;
import chemaxon.struc.Molecule;

/**
 * Simple example of building a molecule from a SMILES string.
 *
 * @author Andras Volford, Miklos Vargyas
 *
 */

public class ReadMoleculeString {
    public static void main(String[] args) {

        try {
             String smiles = "CC>>CC";

             Molecule m = MolImporter.importMol(smiles);

             System.out.println(m.getAtomCount());
             System.out.println(m.toFormat("mol"));

        }
        catch (MolFormatException e) {
             System.err.println("Format not recognised.");
         }
    }
}
```

For a complete source code example with a simple GUI, please see ImportMoleculeSource.java.

Importing from InputStream

Importing one molecule where the molecule source is available via an `InputStream`:

```
try {
        MolImporter mi = new MolImporter(stream);
    Molecule mol = mi.read();
    // do something with the molecule
    mi.close();
} catch (IOException e) {
    // handle the exception
}
```

For a complete source code example with a simple GUI, please see ImportFromStream.java.

Please note that the MolImporter needs to be closed explicitly!

Importing options

Additional options of MolImporter allow to refine behavior further. Options can be general or dependent on file formats. Options can be set in the constructor `MolImporter(InputStream is, String opts)` or during import with static method `MolImporter.importMol(String s, String opts)`. The most important option is the file format option which specifies the format to read from. However, without this format option the automatic format recognition will detect the format. If the import speed is an important factor then the format option is strongly recommended. General or file format dependent options are separated by a colon from the file format option and by a comma from each other. In the code example below the molecules from the imported multi-structure file are merged into one structure with the `MULTISET` option. This is a general option and can be applied in case of any file format. File format option is `sdf` for MDL MOL SD file format and file format specific option is `Usg` to ungroup any S-group found in the structure file.

```
try {
    MolImporter importer = new MolImporter(new FileInputStream("examples/io/basic/mols.sdf"), "sdf:MULTISET,
Usg");
    Molecule molecule = importer.read();
    // do something with the molecule
    mi.close();
} catch (IOException e) {
    // handle the exception
}
```

For a complete source code, please see ImportExportOptions.java.

Note that after importing SMILES, invoking of `MoleculeGraph.clearCashedInfo` method is recommended in order to remove cashed information which results increased molecule size.

Importing a multi-molecule file

Importing molecules from a multi-molecule file given with URL:

```
try {
    URL url = new URL(path);
    MolImporter importer = new MolImporter(url.openStream());
    Molecule mol;
    while ((mol = importer.read()) != null) {
            // do something with the molecule
    }
    importer.close();
} catch (IOException e) {
    // handle the exception
}
```

Working code example of importing molecules from file

```
/*
 * Copyright (c) 1998-2014 ChemAxon Ltd. All Rights Reserved.
 */
import java.io.IOException;
import chemaxon.formats.MolFormatException;
import chemaxon.formats.MolImporter;
import chemaxon.struc.Molecule;
/**
 * Example class for molecule import.
 *
 * @author Andras Volford, Miklos Vargyas
 *
 */
public class ReadMoleculeFile {

    public static void main(String[] args) {

        String filename = args[0];

        try {
            // create a molecule importer for the given file
            MolImporter mi = new MolImporter(filename);

            // read the first molecule from the file
            Molecule m = mi.read();

            while (m != null) {
                printProperties( m );
                // read the next molecule from the input file
                m = mi.read();
             }
            mi.close();
         }
        catch (MolFormatException e) {
             System.err.println("Molecule format not recognised.");
         }
        catch (IOException e) {
             System.err.println("I/O error:" + e);
         }
    }

    private static void printProperties( Molecule m ) {
         System.out.println( "\nMolecule " + m.getName() );
         int nProps = m.getPropertyCount();
         for ( int i = 0; i < nProps; i++ ) {
             String propKey = m.getPropertyKey( i );
             String propValue = m.getProperty( propKey );
             System.out.println( propKey + " = " + propValue );
         }
    }
}
```

For a complete source code example with a simple GUI, please see ImportMultiMoleculeFile.java .

Iterating on molecules where the molecule is the target of the "foreach" statement:

```
URL url = new URL("http://www.chemaxon.com/marvin/mols-2d/mols.sdf");
for (Molecule molecule : new MolImporter(url.openStream())) {
     // do something with the molecule
}
```

Please note that only one Iterator per MolImporter is working at the moment. For a complete source code, please see ImportIterator.java.

Accessing a molecule directly

A molecule in a multi-molecule input can be accessed directly, seeking to the second molecule in this code example:

```
URL url = new URL("http://www.chemaxon.com/marvin/mols-2d/mols.sdf");
MolImporter importer = new MolImporter(url.openStream(), "sdf");
importer.seekRecord(2, null);
Molecule molecule = importer.read();
```

For a complete source code, please see SeekingMolecule.java, `seekRecord` method.

Importing with MRecordImporter

Iterating on molecules and documents is possible with the `MRecordImporter` class also:

```
try {
        URL url = new URL(path);
    MolInputStream mis = new MolInputStream(url.openStream(), null, null, null);
    MRecordImporter importer = new MRecordImporter(mis, null);
    MDocument mDocument;
    while ((mDocument = importer.readDoc()) != null) {
        Molecule mol = (Molecule) mDocument.getMainMoleculeGraph();
        // do something with the molecule
    }
    importer.close();
} catch (MRecordParseException ex) {
    ex.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}
```

For a complete source code, please see ImportMultiMoleculeFile.java, `importMoleculeWithMRecordImporter` method.

Importing with MRecordReader

Iterating on records only is possible with the `MRecordReader` class :

```
MRecordReader recordReader = MFileFormatUtil.createRecordReader(
                                new FileInputStream(new File("examples/io/basic/mols.rdf")),
                                null, null, null);
MRecord record = null;
int recordCount = 0;
while ((record = recordReader.nextRecord()) != null) {
        // do something with the record
        recordCount++;
}
```

For a complete source code, please see ImportRecords.java.