

Bachelor Thesis

von

Patrick Pfeifer

Programmierung einer Auswertsoftware für RNA-seq Daten

Professor:

Prof. Dr. Georg Lipps

Experte:

Dr. Schuierer



Fachhochschule Nordwestschweiz
Hochschule für Life Sciences

Institut für Chemie und Bioanalytik

Muttenz, Juli 2012

Contents

1	Einleitung	3
1.1	Zusammenfassung	3
1.2	Motivation / Ziel	3
1.3	Theoretischer Hintergrund	3
2	Architektur der implementierten Lösung	7
2.1	Komponenten	7
2.2	Technologie-Umfeld	7
2.3	Begründung von Design und Technologieauswahl	7
2.4	Erfahrungsbericht	7
3	Die Software	7
3.1	Screenshots	7
3.2	Quellcode	7
3.2.1	Package rnaseqlyze	7
3.2.2	Package rnaseqlyze.core	27
3.2.3	Package rnaseqlyze.cli	40
3.2.4	Package rnaseqlyze.web	49
3.2.5	Package rnaseqlyze.worker	60
3.2.6	Javascript Code	76
4	Ausblick	87
4.1	Komplettierung	87
4.2	Inbetriebnahme	87
5	Anhang	87
5.1	Glossar	87

Todo list

weitere Beispiele ?	3
od. "unter anderem" ?	3

1 Einleitung

1.1 Zusammenfassung

Die Vorliegende Arbeit beschreibt die Architektur und die Anwendungsmöglichkeiten einer neuen Software zur Analyse von RNA-seq Daten.

1.2 Motivation / Ziel

Die Erforschung der Funktionsweise prokaryontischer Lebensformen birgt ein enormes Potential. Bereits heute sind zahlreiche Industriezweige von der Kultivierung biotechnologisch hergestellter prokaryontischer Lebewesen abhängig.

- Bsp. Diovon Herstellung
- Bsp. Kläranlage
- Bsp. ...

weitere
Beispiele
?

Aufgrund der grösseren Popularität und daher auch besseren Anwendungs- und Vermarktungsmöglichkeiten Theoretischer Genforschung an Eukaryonten, konzentrieren sich die Anstrengungen vieler Forscher heute aber auf andere Schwerpunkte.

Aufgrund dessen werden daher heute verhältnismässig wenige Studien auf diesem Gebiet durchgeführt. Ausserdem, stehen die Auswertung der durch diese Studien produzierten Daten nur wenige, bzw. keine brauchbaren Werkzeuge, d.h. Web- oder Desktop- Software, zur Verfügung.

Die Herstellung einer solchen Software im Rahmen einer Bachelor-Thesis bietet sich an. Zum einen erhalte ich als Student der Biomedizinischen Informatik einen Einblick in die die Welt der Genforschung. Zum andern kann ich hoffentlich mit meiner Arbeit einen kleinen Beitrag zur Verbesserung der Zugänglichkeit zu der beeindruckenden Menge an RNA-Sequenzdaten leisten.

1.3 Theoretischer Hintergrund

Seit einiger Zeit bieten eine Hanvoll Produzenten Gensequenzierungsanlagen an, die auf Prinzipien beruhen, welche die Kosten pro sequenziertem Basenpaar drastisch verringern konnten. Diese "next-generation" Sequencing Plattformen werden vor allem dazu verwendet, die Transkriptome, also die gesamte in einer Zelle bzw. einem einzelligen Organismus vorhandene RNA zu sequenzieren.

od.
"unter
anderem"
?

Gegenüber der klassischen Sequenzierung mit Microarrays, bietet RNA-seq verschieden Vorteile, wie z.B. die annotations-unabhängige Messung der Transkription.

verschiedene

Die Erforschung von kompletten prokaryontischen Genomen anhand von RNA-seq hat

birgt heute Studien . Der Fokus verschiebt sich, weg von

RNA-seq ist

Genome-wide studies of bacterial gene expression are shifting from microarray technology to second generation sequencing platforms.

RNA-seq has a number of advantages over hybridization-based techniques, such as annotation-independent detection of transcription, improved sensitivity and increased dynamic range.

Early studies have uncovered a wealth of novel coding sequences and non-coding RNA, and are revealing a transcriptional landscape that increasingly mirrors that of eukaryotes.

Already basic RNA-seq protocols have been improved and adapted to looking at particular aspects of RNA biology, often with an emphasis on non-coding RNAs, and further refinements to current techniques will improve our understanding of gene expression, and genome content, in the future. Go to: Introduction

The advent of second generation sequencing technologies has created many opportunities to improve functional genomics experiments, including quantitative gene expression studies. Most previous transcriptional analysis methods have relied on hybridization of targeted oligonucleotides to particular loci for their sequence specificity: either primers binding to target cDNA in quantitative reverse transcription polymerase chain reaction (qRT-PCR), labeled probes binding to RNA in Northern blotting or hybridization of cDNA to probes on microarray chips. RNA-seq is different in principle in that data are matched to genes by sequence alignment instead.

This has intrinsic advantages: first, because no probe sequences are specified, all transcription is studied in an unbiased manner, and experimental design does not need to be altered in accordance with differences in genome sequence. This promises to be a particular advantage in the study of bacteria with large amounts of genetic variation between strains [1]. It also allows the discovery of novel genetic features, as well as permitting the delineation of operons and untranslated regions, allowing the improvement and extension of sequence annotation.

Second, mapping of sequence data is more precise than hybridization between oligonucleotides. This allows transcription to be studied at a much higher resolution by sequencing, thereby also permitting the study of more repetitive regions of the genome. Additionally, it means quantification of gene expression by RNA-seq does not suffer from the issues of interference between genes due to non-specific hybridization of cDNA to probes [2,3].

Third, whereas hybridization-based methods measure gene expression levels through detection of fluorescence or radioactivity, RNA-seq uses the amount of data matching a given coding sequence (CDS), typically quantified as reads per kilobase CDS length per million reads analyzed (RPKM) [4]. This measure cannot be saturated in the way the detection of light or radioactivity can, hence RNA-seq has a much greater dynamic range for measuring variability in expression levels. Consequently, it can also be much more discriminatory both at high levels of gene

expression and more sensitive at very low levels of expression, given sufficient sequencing depth. Go to: Preparation of cDNA

RNA is typically extracted using organic solvents or commercially available kits; however, care should be taken to ensure the method does not bias the sampling of the transcriptome [5] and is capable of harvesting sufficient starting material needed to construct a sequencing library, as more RNA is typically needed than for microarray experiments. Furthermore, the exclusion of highly expressed transcripts, which risk saturating the dataset, is also more difficult than with microarray experiments, where probes can be omitted from the chip design as required. As ribosomal RNA comprises the vast majority of the extracted RNA population, depletion of these molecules through hybridization to magnetic bead-linked complementary oligonucleotides [5–10,11•], or the use of terminator exonucleases that specifically degrade transcripts with a 5′ monophosphate group [12••], has been used in efforts to increase the coverage of mRNA and ncRNA. However, the rapid increase in the productivity of the second generation sequencing technologies renders the expensive depletion processes largely unnecessary, especially given the opportunity for sample degradation and bias it presents [10]. Nevertheless, saturation of sequence data by abundant transcripts will remain an issue in some cases; for instance, when analyzing bacterial gene expression within host tissues, where eukaryotic RNA will be far more abundant than that of the prokaryote.

In the original RNA-seq protocols, following extensive DNase treatment, RNA was typically converted into cDNA through random hexamer-primed reverse transcription followed by second DNA strand synthesis [5–9,13•]. However, using double stranded cDNA for making sequencing libraries results in equal levels of signal on both the sense and antisense strands, thereby losing information regarding the direction of transcription. A simple method for maintaining the directional signal in RNA-seq data is to construct Illumina libraries from only first strand cDNA [10]. Alternative techniques used to maintain directional fidelity involve sequentially ligating adapters onto RNA molecules in an orientation-specific manner [14,15], with one approach implemented in studies of *Mycoplasma pneumoniae* and *Pseudomonas syringae* transcriptomes [16••,17•] and another used for RNA-seq in *Helicobacter pylori* and *Salmonella enterica* Typhimurium [12••,18•] (Figure 1). Other methods for maintaining directional information pioneered in studies of eukaryotes include the use of template switching PCR [19], bisulfite-induced conversion of cytosine to uracil in transcripts before reverse transcription [20], addition of sequence tags into the primers used for reverse transcription [21] and incorporation of deoxyuridine into the second strand of cDNA, which can subsequently be degraded using uracil-N-glycosylase [22]. The importance of this information in characterizing ncRNA and observing antisense transcription is becoming increasingly evident. Figure 1

Figure 1 Methods for preparation of cDNA. All methods require the extraction of nucleic acids from a sample of cells, followed by the enzymatic removal of DNA. Ribosomal RNA may then be depleted to increase the sequence coverage of other transcripts. To identify (more ...) Go to: Alternative applications of RNA-seq

As well as surveying the entire transcriptomes of bacterial strains, RNA-seq can be adapted to other experiments as well. For instance, techniques have been developed to specifically sequence the 5′ region of RNA molecules, allowing the

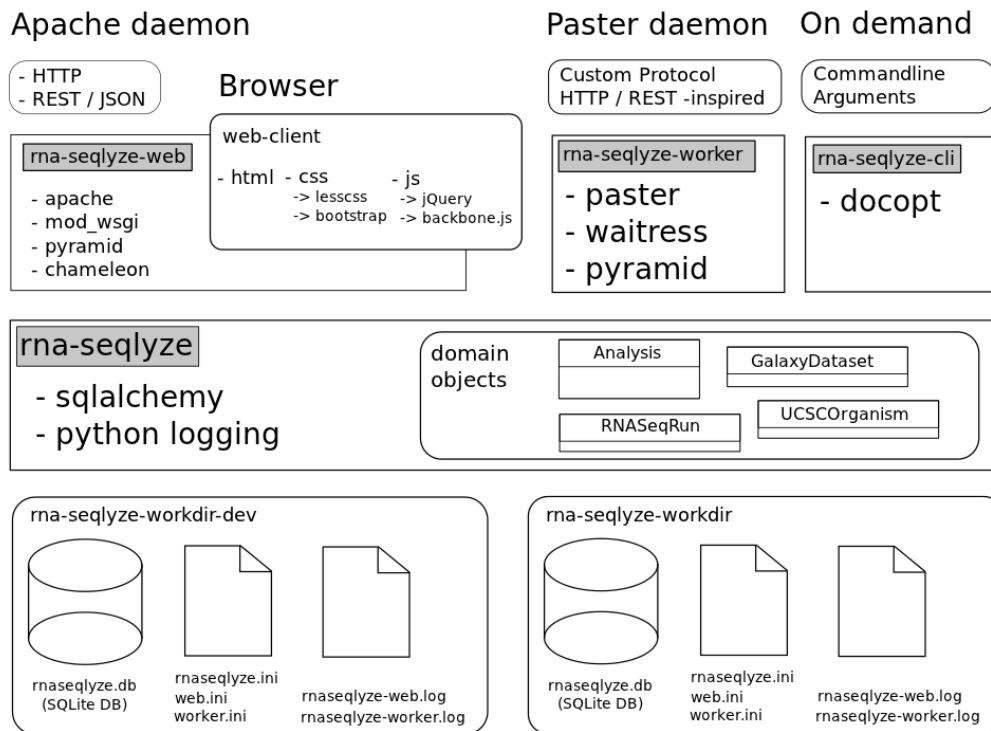


Abb. 1: Übersicht der Architektur der RNA-Seqlyze Applikation. Zuunterst die Konfigurations- und Datenbank- Dateien. In der Mitte die Kernfunktionalität, mit SQLAlchemy als ORM-Mapper. Darüber die einzelnen Schnittstellen. Dann die von den Schnittstellen verwendeten Protokolle. Zuoberst die den Code ausführenden Prozesse.

identification of putative transcriptional start sites and helping to define operons and ncRNA [12••,13•] (Figure 1). In *S. Typhimurium*, coimmunoprecipitation of RNA molecules with Hfq, a chaperone that facilitates hybridization between ncRNA and mRNA, was used to enrich a sample for transcripts participating in such interactions [18•], while in *Vibrio cholerae*, a very stringent depletion and size-selection process was used to specifically sequence small ncRNA [23]. RNA-seq has also been applied to whole environments, leading to the development of techniques for sampling the metatranscriptomes of marine [24,25] and soil communities [26].

2 Architektur der implementierten Lösung

2.1 Komponenten

2.2 Technologie-Umfeld

2.3 Begründung von Design und Technologieauswahl

2.4 Erfahrungsbericht

3 Die Software

3.1 Screenshots

3.2 Quellcode

3.2.1 Package rnaseqlyze

Modul rnaseqlyze

```
"""
Top level package module

Importing this package configures the python "logging"
module in a way that messages of any level go to sys.stderr.
"""

#: the project base-name
#: the -cli, -web and -worker project names
#: are constructed by appending the part name to this one
project_name = "rna-seqlyze"

import pkg_resources
#: The __version__ property is set automatically set to the value of
#: pkg_resources.get_distribution(project_name).version on module import time.
try:
    __version__ = pkg_resources.get_distribution(project_name).version
except:
    # When the packe is initially installed, this module is imported by setup.py
    # and the the `project_name` attribute defined above is used to set the
    # project name, in order to make that information non-redundant.
    #
    # The version, however, is determined at install/build time by running
    # `git --describe` - in setup.py. So it doesn't matter if it is not set at
    # that time. Later on, because `python setup.py install/develop` writes it
    # to rna-seqlyze.egg-info/PKG-INFO, where pkg_resources picks it up,
    # it will be available whenever the package is imported.
    pass
del pkg_resources

import logging
```

```

logging.basicConfig(level=0, format="%(levelname)-5.5s [%(name)s] %(message)s")
del logging

35 def configure(_workdir):
    """
        Calling this function

        - sets rnaseqlyze.workdir to <workdir>

40     - sets rnaseqlyze.<setting> attributes for all
        settings under [rnaseqlyze] in '<workdir>/rnaseqlyze.ini'.

        - imports Bio.Entrez and sets Bio.Entrez.email to rnaseqlyze.admin_email
45     """

    global workdir
    workdir = _workdir

50     from os.path import join
    from ConfigParser import ConfigParser
    config = ConfigParser(dict(here=workdir))
    config.read(join(workdir, 'rnaseqlyze.ini'))

55     for name, value in config.items("rnaseqlyze"):
        globals()[name] = value

    import Bio.Entrez
    Bio.Entrez.email = admin_email

```

Modul rnaseqlyze.build

```

"""
The rna-seqlyze software consisty of several parts. The majority of those parts
have been developped independently of this project and have been released under
a permissive license that allows them to be used in other (permissive licenced)
5 projects like this one.

This file defines a simple system and stores the commands necessary to build and
install those third-party components.
"""

10 from __future__ import print_function

import os, sys, shutil
from os import environ as env
15 from types import MethodType
import subprocess, multiprocessing

# a bit of infrastructure
#####

20 class PartType(type):

    def __init__(cls, *ign):

```



```

25     """
    auto-create and stock instances
    upon creation of "Part" (sub)classes

    appends a new instance of the
    created class to the "parts" list, if it exists
30     """
    try:
        parts.append(cls())
    except NameError:
        pass

35 class Part(object):

    __metaclass__ = PartType

40     def __init__(self):
        self.name = self.__class__.__name__
        try:
            self.subdir = "src/" + self.srcdir
        except AttributeError:
45             self.subdir = "src/" + self.name

    def execute(self, phase):
        cmds = getattr(self, phase, None)
        if cmds == None: return
50         print("#" * 80)
        print("# executing %s '%s' phase" % (self.name, phase))
        print("#")
        dev_null = file("/dev/null")
        logdir = "report/buildlogs"
55         if not os.path.isdir(logdir): os.mkdir(logdir)
        logpath = logdir + "/%s-%s.log" % (self.name, phase)
        T = subprocess.Popen(["tee", logpath], stdin=subprocess.PIPE)
        try:
            import time
60             log = lambda msg="": print(msg, file=T.stdin)
            log(time.asctime())
            log()
            log("\n".join("%s=%s" % nv
65                          for nv in filter(lambda i: i[0] in (
                              "TOPDIR", "PREFIX", "BINDIR", "LIBDIR",
                              "MACHTYPE", "NCPUS_ONLN"), env.iteritems()))
            log()
            if type(cmds) not in (list, tuple):
                cmds = cmds, # make it a 1-tuple
70             for cmd in cmds:
                log("$ cd " + self.subdir)
                log("$ " + "\n ".join(str(cmd).split("\n")))
                log()
                if type(cmd) == str:
75                     ret = subprocess.call(cmd, shell=True, cwd=self.subdir,
                                                stdin=dev_null, stdout=T.stdin, stderr=T.stdin)
                elif type(cmd) == MethodType:

```

```

        def tgt():
            sys.stdin = dev_null
            sys.stdout = sys.stderr = T.stdin
            os.chdir(self.subdir)
            return cmd()
        sp = multiprocessing.Process(target=tgt)
        sp.start()
        sp.join()
        ret = sp.exitcode
    else:
        raise Exception("Invalid '%s' phase command: %s" % (
            phase, repr(cmd)))

    log()
    log(time.asctime())
    if ret != 0:
        raise Exception("%s '%s' phase failed -- exit code %d" % (
            self.name, phase, ret))

    finally:
        T.stdin.close()
        T.wait()

# parts & phases
#####

parts = []
phases = 'build', 'test', 'install'

class bcbcb(Part):
    srcdir = "bcbcb/nextgen"
    build = "python setup.py build"
    # save some time
    #test = "nosetests"
    install = "python setup.py install --prefix=$PREFIX"

class biopython(Part):
    build = "python setup.py build"
    # save some time
    #test = "python setup.py test"
    install = "python setup.py install --prefix=$PREFIX"

class bowtie2(Part):
    build = "make -j$NCPUS_ONLN"
    def install(self):
        """
        the bowtie2 install function
        exists because there is no 'install' target in
        the makefile, so the binaries need to be installed manually
        """
        import shutil
        for f in ("bowtie2" + x for x in ("", "-align", "-build", "-inspect")):
            shutil.copy(f, env["BINDIR"])
            os.chmod(env["BINDIR"] + "/" + f, 0775)

class ncurses(Part):

```

```

    build = "./configure --prefix $HOME/.local && make"
    install = "make install"

135 class samtools(Part):
    depends = ncurses
    build = (
        'make -j$NCPUS_ONLN -C bcftools',
        'make -j2 SUBDIRS=.'
140         ' LIBPATH=-L$PREFIX/lib LIBCURLS=-lncurses'
        ' CFLAGS="$(echo -I$PREFIX/include{,/ncurses})"'
    )
    install = "cp samtools $PREFIX/bin"

145 class cufflinks(Part):
    depends = samtools
    build = "./configure --prefix=$PREFIX" \
        " --with-eigen=$TOPDIR/src/eigen" \
        " --with-bam=$TOPDIR/src/samtools && make"
150    install = "make install"

class kent(Part):
    build = "make -C src/lib"
    def install(self):
155         """
        the kent install function was created, because rna-seqlyze need only
        a small subset of the included utilities and the easiest way to build
        those is to run "make" with custom arguments for each one of them
        """
160         for util in "wigToBigWig bedToBigBed".split(" "):
            if subprocess.call("make -C src/utills/" + util, shell=True) != 0:
                raise Exception("kent.install(): couldn't install '%s'" % util)

class pysam(Part):
165     build = "python setup.py build"
    # tests failing...
    #test = "cd tests; nosetests --exe"
    #test = "cd tests; ./pysam_test.py"
    install = "python setup.py install --prefix=$PREFIX"
170

class rna_seqlyze_cli(Part):
    srcdir = "rna-seqlyze-cli"
    build = "python setup.py build"
    test = "python setup.py test"
175    install = "python setup.py develop --prefix=$PREFIX"

class rna_seqlyze_web(Part):
    srcdir = "rna-seqlyze-web"
    build = "python setup.py build"
180    test = "python setup.py test"
    install = "python setup.py develop --prefix=$PREFIX"

class rna_seqlyze_worker(Part):
    srcdir = "rna-seqlyze-worker"
185    build = "python setup.py build"

```

```

test = "python setup.py test"
install = "python setup.py develop --prefix=$PREFIX"

class sra_sdk(Part):
190     # To get this to compile, I
    # 1) created a symlink src/sra_sdk/libxml2.so
    # pointing to /usr/lib/libxml2.so.2 and added
    # LDFLAGS=-L$PWD to avoid having to install libxml2-dev
    # 2) replaced the content of src/sra_sdk/libs/ext/Makefile
195     # with "all:" to skip unnesessary downloading of zlib and libbz2
    build = "LD_RUN_PATH=$LIBDIR make STATIC= STATICSYSLIBS= LDFLAGS=-L$PWD"
    install = (
        "cp -a linux/pub/gcc/$ARCH/bin/* $BINDIR",
        "cp -a linux/pub/gcc/$ARCH/lib/* $LIBDIR",
200     "cp -a linux/pub/gcc/$ARCH/mod $LIBDIR/ncbi",
        "cp -a linux/pub/gcc/$ARCH/wmod $LIBDIR/ncbi",
    )

class tophat(Part):
205     build = "./configure --prefix=$PREFIX" \
        " --with-bam=$TOPDIR/src/samtools && make"
    install = "make install"

class trac(Part):
210     build = "python setup.py build"
    # save some time
    #test = "python setup.py test"
    install = "python setup.py install --prefix=$PREFIX"

215 class trac_env(Part):
    def install(self):
        # need to discuss server
        # configuration with admin
        #destdir = "%(PREFIX)s/var/trac_env" % env
220     #basedir = os.path.dirname(destdir)
        #if not os.path.isdir(basedir):
        #    os.mkdir(basedir)
        #shutil.copytree(".", destdir, symlinks=True)
        #print("Copied %s to %s\n" % (os.getcwd(), destdir))
225     print("\n".join((
        """\
The following still needs to be done manually:
1) Set up a database
2) Restore the backup:
230     $ cd "" + os.getcwd() + ""
        $ mysql -uUSERNAME -pPASSWORD DATABASE < mysql-db-backup.sql
        4) Adjust the 'database' variable in the [trac] section in 'conf/trac.ini':
        database = mysql://USERNAME:PASSWORD@localhost/DATABASE
        """))
235     )))

class transterm_hp(Part):
    build = "make"
    def install(self):

```

```

240     prog = "transterm"
        data = "expterm.dat"
        shutil.copy(prog, env["BINDIR"])
        os.chmod(env["BINDIR"] + "/" + prog, 0775)
        shutil.copy(data, env["LIBDIR"])

245
class s3cmd(Part):
    install = "python setup.py install --prefix=$PREFIX"

class docopt(Part):
250     install = "python setup.py install --prefix=$PREFIX"

```

Modul rnaseqlyze.efetch

```

from Bio import Entrez

import rnaseqlyze

5  nc_db = "nucore"
    gb_type = "gb"
    gb_mode = "text"

def get_nc_id(accession):
10     handle = Entrez.esearch(db=nc_db, term=accession + "[Accession]")
        id_list = Entrez.read(handle)["IdList"]
        if len(id_list) != 1:
            raise Exception("unexpected reply from Entrez: id_list: %s" % id_list)
        return id_list[0]

15
def fetch_nc_gb(gb_id, out_file):
    handle = Entrez.efetch(db=nc_db, id=gb_id, rettype=gb_type, retmode=gb_mode)
    from shutil import copyfileobj as copy
    copy(handle, out_file)

```

Modul rnaseqlyze.galaxy

```

"""
RNA-Seqlyze Galaxy Module

Shamelessly piggy-back onto Penn-State University's "Galaxy" Project.

5  RNA-Seqlyze needs a some publicly available Web-Space, which PSU provides
    plenty of for bioinformatics research data (250.0 Gb per user as of 4 Jul 2012).

    Thanks go to Penn-State University!

10  http://www.psu.edu/

    """

15  # FIXME: the whole code here needs heavy refactoring

import logging

```

```

log = logging.getLogger(__name__)

20 import os, json, time, ftplib, \
    urllib, urllib2, cookielib
    from threading import local
    from datetime import datetime, timedelta

25 import lxml.html

    import rnaseqlyze
    from rnaseqlyze import multipart

30 email = 'ucgxccgr@mailinator.com'
    password = 'brtbhcdg'
    api_key = 'dddb2c53c96c0c4d263e6c74b507d203'
    hostname = 'main.g2.bx.psu.edu'

35 default_history = '16f9a8e916e0e908'

    default_history_url = 'https://main.g2.bx.psu.edu/u/dcgdfvtcdv/h/rna-seqlyze'

    history_path_template = '/api/histories/{history}/contents'
40 ucsc_bam_track_template = \
    '/display_application/{dataset}/ucsc_bam/archaea/None/param/track'

    ucsc_bam_path_template = \
    '/display_application/{dataset}/' \
45     'ucsc_bam/archaea/None/data/galaxy_{dataset}.bam'

    dataset_info_url_template = "/api/histories/{history}/contents/{dataset}"

    dataset_display_url_template = "/datasets/{dataset}/display"

50 rq_headers = {}

    class Session(local):
        cookies = None
55         created = None

    session = Session()

    def api_call(path):
60         url = "https://" + hostname + path
        return urllib2.urlopen(url + "?key=" + api_key).read()

    def login():
        cookie_jar = cookielib.CookieJar()
65         urllib2.install_opener(urllib2.build_opener(
            urllib2.HTTPCookieProcessor(cookie_jar)))
        log.info("Loggin in to galaxy server %s ..." % hostname)
        login = "https://" + hostname + "/user/login"
        rq = urllib2.Request(login, headers=rq_headers)
70         request = urllib2.urlopen(rq)
        doc = lxml.html.parse(request).getroot()

```

```

form = doc.forms[0]
form.fields["email"] = email
form.fields["password"] = password
75 submit = "login_button", form.fields["login_button"]
data = urllib.urlencode(form.form_values() + [submit])
log.debug("posting login form: %s" % form.action)
rq = urllib2.Request(form.action, headers=rq_headers)
request = urllib2.urlopen(rq, data)
80 doc = lxml.html.parse(request).getroot()
log.info("Success!")
return cookie_jar

def import_upload(filename):
85     if not (session.created
            and session.created > (datetime.now() - timedelta(minutes=30))):
        session.cookies = login()
        session.created = datetime.now()

90     urllib2.install_opener(urllib2.build_opener(
        urllib2.HTTPCookieProcessor(session.cookies)))
    log.info("Importing ftp file")
    tool = "https://" + hostname + "/tool_runner?tool_id=upload1"
    rq = urllib2.Request(tool, headers=rq_headers)
95     request = urllib2.urlopen(rq)
    doc = lxml.html.parse(request).getroot()
    found = False
    form = doc.forms[0]
    inp = form.inputs["files_0|ftp_files"]
100    if isinstance(inp, lxml.html.InputElement):
        if inp.attrib['value'] == filename:
            found = inp.checked = True
    elif isinstance(inp, lxml.html.CheckboxGroup):
        for box in inp:
105             if box.attrib['value'] == filename:
                found = box.checked = True
    else:
        raise Exception("unexpected html element: %s" % inp)
    if not found:
110         raise Exception("file not available for import: %s" % filename)
    submit = "runtool_btn", form.fields["runtool_btn"]
    data = multipart.urlencode(form.form_values() + [submit])
    log.debug("posting upload form: %s" % form.action)
    rq = urllib2.Request(form.action, data, headers=rq_headers)
115    request = multipart.urlopen(rq)
    doc = lxml.html.parse(request).getroot()
    log.info("Success!")

def ftpupload(fileobj, filename):
120     """
    upload a file object to galaxy
    based on http://love-python.blogspot.com/2008/02/ftp-file-upload.html
    """
    log.info("uploading file to ftp server")
125    ftp = ftplib.FTP(hostname, email, password)

```

```

ftp.storbinary('STOR ' + filename, fileobj)
log.info("Success!")
ftp.quit()

130 def upload(fileobj, filename):

    # can't initialize this at module import time
    # because rnaseqlyze.xxx properties not initialized
    global rq_headers
135     try:
        mail = rnaseqlyze.admin_email
    except:
        # rnaseqlyze not .configure()d
        mail = os.getenv("USER") + "@" + os.uname()[1]
140     rq_headers = {
        'User-Agent': "%s (version:%s / admin:%s)" % (
            rnaseqlyze.project_name, rnaseqlyze.__version__, mail),
        # 'User-Agent': "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:13.0)" \
        # " Gecko/20100101 Firefox/13.0.1",
145     }

    ftpupload(fileobj, filename)
    import_upload(filename)
    datasets = json.loads(api_call(
150         history_path_template.format(history=default_history)))
    # assume objects are ordered chronologically...
    for dataset in reversed(datasets):
        if dataset['name'] == filename:
            return dataset['id']
155     raise Exception("Couldn't find id of uploaded file in dataset")

```

Modul rnaseqlyze.gb2ptt

```

import logging
log = logging.getLogger(__name__)

import sys

5
from Bio import SeqIO
from Bio.SeqFeature import ExactPosition

def gb2ptt(in_file, out_file):
10
    debug = False

    seq = SeqIO.parse(in_file, "genbank").next()

15
    # expected input:
    #
    # CDS 249..857
    #     /locus_tag="SSD0001"
    #     /note="Predicted membrane protein, conserved in archaea"
20    #     /codon_start=1

```



```

# /transl_table=11
# /product="hypothetical protein"
# /protein_id="NP_341578.1"
# /db_xref="GI:15896973"
# /db_xref="GeneID:1455258"
# /translation="MITEFLLKKKLEEHLSHVKEENTYVTDLVRCPRRVRYESEYKE
# LAISQVYAPSAILGDILHLGLESVLKGNFNAETEVELREINVGGKVYKIKGRADAI
# RNDNGKSIVIEIKTSRSDKGLPLIHKKMLQIYLWLFSAEKGILVYITPDRIAEYEIN
# EPLDEATIVRLAEDTIMLQNSPRFNWECKYCIFSVCAPAKLT"

# intended output:
#
# Sulfolobus solfataricus P2 chromosome, complete genome - 1..2992245
# 2978 proteins
# Location      Strand  Length  PID
# 249..857      +       202    15896973

# Gene      Synonym Code      COG      Product
# -         SS00001 -      COG1468L      hypothetical protein

import csv
writer = csv.writer(out_file, delimiter='\t', lineterminator='\n')
writer.writerow((seq.description,))
writer.writerow(())
writer.writerow(('Location', 'Strand', 'Length', 'PID',
                  'Gene', 'Synonym', 'Code', 'COG', 'Product'))

n=0
for f in seq.features:
    n+=1
    if debug and n > 10:
        break

    if f.type != 'CDS':
        continue

    if type(f.location.start) != ExactPosition \
    or type(f.location.end) != ExactPosition:
        log.info("skipping non-exact location '%s' in '%s'" % (
            f.location, f.type))
        continue

    _len = f.location.end.position - f.location.start.position
    if _len < 0:
        _len = len(seq.seq) \
            - f.location.start.position + f.location.end.position
    if _len % 3:
        log.info("implausible feature length (%d) in '%s'" % (_len, f.type))
    _len //= 3 # integer division
    _len -= 1 # omit stop codon

    xrefs = dict(map(lambda s: s.split(':'), f.qualifiers['db_xref']))
    for r in 'GI', 'GeneID':

```

```

75         if r not in xrefs:
            xrefs[r] = '-'

        for q in 'gene', 'product', 'locus_tag':
            if q not in f.qualifiers:
80                 f.qualifiers[q] = '-'

        # need to convert between biopython (0-based, incl:excl)
        # and genbank (1-based, incl:incl) position boundary notation here
        writer.writerow((
85             "%d..%d" % (f.location.start.position+1, f.location.end.position),
            ['-','+'][(f.strand + 1) / 2],
            _len,
            xrefs['GI'],
            f.qualifiers['gene'][0],
90             f.qualifiers['locus_tag'][0],
            '-',
            '-',
            f.qualifiers['product'][0]
        ))
95
    log.info("wrote %d rows" % n)

```

Modul rnaseqlyze.install

```

"""
RNA-Seqlyze Install

This command builds and installs all software components
5 included with and required by the RNA-Seqlyze web application.

Usage:
    rnas-install
    rnas-install --prefix <dir>
10    rnas-install -h|--help

Note:
    The command has must run from the top level RNA-Seqlyze source directory.

15 Options:
    --prefix <dir>
        The option is passed on to the ./configure and install
        scripts of the various programs that this command installs.
        The effect is, that all produced executables will be
20 installed under the that directory.

    If not specified, defaults to ``$HOME/.local``

.. important::
25     The `PREFIX` variable in the "/etc/init.d/rnaseqlyze.sh"
        worker daemon startup script and the `prefix` variable
        in the "/var/www/./rna-seqlyze.wsgi" script must both
        be set to the directory specified here!
"""

```

```

30 import os, re
   from os import environ as env
   from os.path import join, exists

35 from rnaseqlyze.build import parts, phases

   def main():
       import docopt
       opts = docopt.docopt(__doc__)

40       assert exists("src/rna-seqlyze/rnaseqlyze/__init__.py"), \
           "This command must be run from the top level RNA-Seqlyze source directory!"

       topdir = os.getcwd()
45       prefix = opts['--prefix'] \
           or join(os.getenv("HOME"), ".local")

       env["TOPDIR"] = topdir
       env["PREFIX"] = prefix
50       env["BINDIR"] = prefix + "/bin"
       env["LIBDIR"] = prefix + "/lib"
       env["MACHTYPE"] = os.uname()[4]
       env["ARCH"] = re.sub('i.86', 'i386', env["MACHTYPE"])
       env["NCPUS_ONLN"] = str(os.sysconf("SC_NPROCESSORS_ONLN"))

55       for part in parts:
           for phase in phases:
               part.execute(phase)

60       print "RNA-Seqlyze sucessfully installed."
       print
       print "    PREFIX=%s" % prefix
       print

```

Modul rnaseqlyze.multipart

```

"""
Multipart form-data handling
based on http://code.activestate.com/recipes/146306/
"""

5 import uuid, urllib2, mimetypes

def urlopen(url, data=None):
    if isinstance(url, basestring):
        rq = urllib2.Request(url, data)
10    elif isinstance(url, urllib2.Request):
        rq = url
        data = rq.data
    else:
        raise Exception("'url' parameter must be a string or urllib2.Request")

15    try:
        boundary = data[2:data.index("\r")]

```

```

except ValueError, e:
    raise Exception("couldn't find boundary string in data", e)
20 rq.add_header('Content-Type', 'multipart/form-data; boundary=%s' % boundary)
    return urllib2.urlopen(rq)

def urlencode(fields, files=None):
    """
    25 :param asd:
        is a sequence of ``(name, value)`` elements for regular form fields.

    :param files:
        is a sequence of ``(name, filename, value)``
    30 elements for data to be uploaded as files

    :returns:
        ``str`` of multipart/form-data encoded fields + files
    """
    35 boundary = str(uuid.uuid4())
    data = []
    for (key, value) in fields:
        data.append('--' + boundary)
        data.append('Content-Disposition: form-data; name="%s"' % key)
    40 data.append('')
        data.append(value)
    if files:
        for (key, filename, value) in files:
            data.append('--' + boundary)
            data.append('Content-Disposition: form-data' \
    45                 '; name="%s"; filename="%s"' % (key, filename))
            data.append('Content-Type: %s' % get_content_type(filename))
            data.append('')
            data.append(value)
    50 data.append('--' + boundary + '--')
    data.append('')

    return '\r\n'.join(data)

55 def get_content_type(filename):
    return mimetypes.guess_type(filename)[0] or 'application/octet-stream'

```

Modul rnaseqlyze.org.cache

```

"""
    RNA-Seqlyze keeps a cache of organisms available
    in the UCSC Browser. In addition to that, for each of
    those organisms, the matching refseq accession is cached.
    5 """
    import logging
    log = logging.getLogger(__name__)

    import csv
    10 import difflib
    from pkg_resources import resource_stream

```

```

from Bio import Entrez

15 from rnaseqlyze import ucscbrowser
from rnaseqlyze.core.entities import UCSCOrganism

prokaryotes_tsv = "refseq-data/prokaryotes.txt"

20 def refresh(db_session):
    """
    Refresh the organism cache.

    The cache is initialized from the list of organisms available in the
25 UCSC genome browser. A list of rnaseqlyze.orm.UCSCOrganism's is
    retrieved by calling rnaseqlyze.ucscbrowser.get_org_list().

    The retrieved list is not ready to be .add()ed to the :param:db_session
    however, because the objects' primary keys, the refseq accession,
30 are still missing.

    Those are determined by parsing the list of complete genomes available in
    the ncbi "genome" database, which is stored in

35 rnaseqlyze/refseq-data/prokaryotes.txt

    The file was retrieved from

    ftp://ftp.ncbi.nih.gov/genomes/GENOME_REPORTS/prokaryotes.txt
40 on Mon, 02 Jul 2012.

    Once found, the rnaseqlyze.orm.UCSCOrganism objects are updated with the
    refseq accessions and .add()ed to the passed :param:db_session.
45 """

    organisms = ucscbrowser.get_org_list()
    accessions = get_accessions()

50 for org in organisms:
    ot = org.title
    for gt, acc in accessions:
        if ot == gt:
            org.acc = acc
55 db_session.add(org)

    not_found = set(organisms) - \
        set(db_session.query(UCSCOrganism).all())

60 # try fuzzy-matching the title
# of those organisms that were not found
for org in not_found:
    ot = org.title
    best_ratio = 0
65 best_match = None
    for gt, acc in accessions:

```

```

ratio = difflib.SequenceMatcher(None, ot, gt).ratio()
if ratio > best_ratio:
    best_ratio = ratio
    best_match = acc
    best_match_t = gt

if best_ratio > 0.8:
    if db_session.query(UCSCOrganism).get(best_match):
        log.debug(("NOT using '{match}' for '{org}'"
                    " despite match ratio of: {ratio}").format(
                        match=best_match_t, org=ot, ratio=best_ratio))
    else:
        log.info(("using '{match}' for '{org}'"
                  " match ratio: {ratio}").format(
                      match=best_match_t, org=ot, ratio=best_ratio))
        org.acc = best_match
        db_session.add(org)
    else:
        log.warn(("'{org}' not found in NCBI 'genome' database"
                  " (best match ratio only {ratio})").format(
                      org=ot, ratio=best_ratio))

# make sure that that regular expression in views.post() that translates
# the 'org_accession' from 'title (db/accession)', as generated in
# rnaseqlyze.create.js, back to 'accession' doesn't fail
for org in db_session.query(UCSCOrganism).all():
    if any(needle in heystack
            for needle in '()'
            for heystack in (org.db, org.acc, org.title)):
        log.warn("Dropping organism with parenthesis"
                  " to avoid problems in parsing auto"
                  " completed form input in views.post()")
        db_session.expunge(org)

def get_accessions():

    data_file = resource_stream(__name__, prokaryotes_tsv)
    reader = csv.reader(data_file, delimiter='\t')
    headings = reader.next()
    colnums = dict(zip(headings, map(headings.index, headings)))
    # -> { '#Organism/Name': 0, ..., 'Chromosomes/RefSeq': 7 }
    ret = []
    for cols in reader:
        if cols[colnums['Chromosomes/RefSeq']] == '-':
            continue
        ret.append((cols[colnums['#Organism/Name']],
                    cols[colnums['Chromosomes/RefSeq']]))
    return ret

```

Modul rnaseqlyze.s3

```

"""
Upload files to Amazon S* using the s3cmd tools

```

```

The access credentials must be configured in ~/.s3cfg .
5 The s3cmd creates this file interactively with the --configure option.
   """

   #: The s3 bucket name in "S3Uri" format
   base_uri = "s3://biocalc/"

10 import os

   # import this to fix a circular import dependency problem in s3cmd ...
   import S3.Exceptions

15 from S3.S3 import S3
   from S3.Config import Config
   from S3.S3Uri import S3Uri
   from S3.SortedDict import SortedDict

20 # same "interface" like rnaseqlyze.galaxy
   def upload(fileobj, filename):

       cfg = Config()

25       cfg.read_config_file(os.path.join(os.getenv("HOME"), ".s3cfg"))
       cfg.progress_meter = False
       cfg.acl_public = True

30       s3 = S3(cfg)

       headers = SortedDict(ignore_case = True)
       headers["x-amz-acl"] = "public-read"
       headers["x-amz-storage-class"] = "REDUCED_REDUNDANCY"

35       remote_uri = S3Uri(base_uri + filename)

       fileobj.seek(0,2) # seek to end
       size = fileobj.tell()
       fileobj.seek(0) # seek to start

40       response = s3.send_file_multipart(fileobj, headers, remote_uri, size)

       assert response['status'] == 200

45       return remote_uri.public_url()

```

Modul rnaseqlyze.transterm

```

"""
A module to run transterm_hp
"""

5 import os, subprocess
   from subprocess import PIPE

```

```

def run(args, out=None, err=None):
    """
10    Run transterm_hp with the given arguments plus "-p expterm.dat"
    """
    def findit():
        for path in os.getenv("PATH").split(os.path.pathsep):
            for name in os.listdir(os.path.join(path, "../lib")):
15                if name == 'expterm.dat':
                    return os.path.join(path, "../lib", name)
    expterm_dat = findit()
    if not expterm_dat:
        raise Exception("'expterm.dat' not found")
20    cmd = ('transterm', '-p', expterm_dat) + tuple(args)
    proc = subprocess.Popen(cmd, stdout=out, stderr=err)
    proc.wait()
    if proc.returncode != 0:
        raise Exception(str(cmd) + " failed")
25
def tt2bed(tt_output, bed_file):
    for id, begin, end, strand, confidence in iterator(tt_output):
        # let color vary from 0 (black) to 100 (gray)
        rgb_color = ','.join((str(100 - int(confidence)),)*3)
30    print >> bed_file, '\t'.join((
        'chr', begin, end, 'TERM_' + id,
        str(confidence), strand, begin, end, rgb_color
    ))

35 def iterator(tt_output):
    for line in tt_output:
        if not line.startswith(" TERM"):
            continue
        TERM, id, begin, dash, end, \
40         strand, position, confidence, rest = line.split(None, 8)
        # switch begin & end on reverse strand
        if strand == '-':
            begin, end = end, begin
        yield id, begin, end, strand, int(confidence)

```

Modul rnaseqlyze.ucscbrowser

```

"""
Tools to deal with the UCSC genome browser at http://archaea.ucsc.edu/
"""
import logging
5 log = logging.getLogger(__name__)

from json import load
from urllib2 import urlopen
from urlparse import urljoin
10 from StringIO import StringIO
from shutil import copyfileobj
from os import listdir, makedirs
from os.path import join, dirname, isdir

```



```

15 from lxml.html import parse
   from lxml.etree import dump

   import rnaseqlyze
   from rnaseqlyze.core import security
20 # delay import because of a
   # circular import dependency ...
   #from rnaseqlyze.core.entities import UCSCOrganism

   cart_reset_url = "http://archaea.ucsc.edu/cgi-bin/cartReset"
25 custom_track_url = "http://archaea.ucsc.edu/cgi-bin/hgTracks"
   custom_track_params = "?db={org_db}&hgt.customText={track_url}"

   class BigDataTrack(str):
       """
30     A UCSC "Big Data Track"

       You should pass a 'name' and a 'url'
       keyword argument to the constructor.
       """

35     template = 'track type="{type}" name="{name}" bigDataUrl="{url}"'

       __new__ = lambda cls, **kwargs: cls.template.format(type=cls.type, **kwargs)

40 class BAMTrack(BigDataTrack): type = "bam"
   class BigWigTrack(BigDataTrack): type = "bigWig"
   class BigBedTrack(BigDataTrack): type = "bigBed"

   # FIXME:
45 #     The org_list_default_dir = dirname(__file__)
   #     hack will not work if the distribution is installed
   #     as a zipped .egg. pkg_resources.resource_stream or
   #     pkg_resources.resource_string should be used instead.
   org_list_base_url = "http://archaea.ucsc.edu/wp-content/data/"
50 org_list_default_dir = join(dirname(__file__), "ucscbrowser-data")
   json_links_file_name = "ucsc-wp-data.html"

   def get_org_list():
       global UCSCOrganism
55   from rnaseqlyze.core.entities import UCSCOrganism

       global org_list_cache_dir
       if not hasattr(rnaseqlyze, 'ucsc_org_list_cache_dir'):
           raise Exception("rnaseqlyze.configure(workdir) "
60                           "must be called before calling this function")
       org_list_cache_dir = rnaseqlyze.ucsc_org_list_cache_dir

       if not isdir(org_list_cache_dir):
           makedirs(org_list_cache_dir)

65   orgs = []
       for org in get_organisms(get_json_files()):
           for existing in orgs:

```

```

        if existing.title == org.title:
70             log.warn("'%' already present (db: %s/%s)" % \
                        (org.title, org.db, existing.db))

                break
        else:
            orgs.append(org)
75     return orgs

def get_json_files():

    json_links_file = None
80     json_files = None

    for get_json_links_file in (get_json_links_file_web,
                                get_json_links_file_cache,
                                get_json_links_file_default,):
85         try:
            log.debug("trying %s" % get_json_links_file.func_name)
            json_links_file = get_json_links_file()
        except Exception, e:
            log.warn("%s failed: %r" % (get_json_links_file.func_name, e))
90             continue

    for get_json_files in (get_get_json_files_web(json_links_file),
                           get_json_files_cache,
                           get_json_files_default,):
95         try:
            log.debug("trying %s" % get_json_files.func_name)
            return get_json_files()
        except Exception, e:
100             log.warn("%s failed: %r" % (get_json_files.func_name, e))

    raise Exception("Couldn't get json organism lists")

# getting the links file
105
def get_json_links_file_web():
    json_links_file = StringIO()
    copyfileobj(urlopen(org_list_base_url), json_links_file)
    json_links_file.seek(0)
110     return json_links_file

def get_json_links_file_cache():
    return open(join(org_list_cache_dir, json_links_file_name))

115 def get_json_links_file_default():
    return open(join(org_list_default_dir, json_links_file_name))

# getting json files
120
def get_get_json_files_web(json_links_file):

```

```

def get_json_files_web():
    for e in parse(json_links_file).getroot().iter("a"):
125         link = e.attrib['href']
        if link.endswith(".json"):
            security.check_valid_filename(link)
            # FIXME: The json files should also be returned as StringIO
            #         buffers only and the cache files shouldn't be
130             #         overwritten until it is certain that the newly
            #         downloaded files contain the expected data
            json_file = open(join(org_list_cache_dir, link), "w+")
            copyfileobj(urlopen(urljoin(
                org_list_base_url, link)), json_file)
135             json_file.seek(0)
            yield json_file

    try:
        json_links_file.fileno()
140         # json_links_file was defaults or cached
    except:
        # json_links_file was a memory buffer - save it because
        # if this code is reached, it means there was no error,
        # in the code above, so the buffer likely contains good links
145         json_links_file.seek(0)
        copyfileobj(json_links_file, open(join(
            org_list_cache_dir, json_links_file_name), "w"))

    return get_json_files_web

150
def get_json_files_cache():
    for json in listdir(org_list_cache_dir):
        if json.endswith(".json"):
            yield open(join(org_list_cache_dir, json))
155
def get_json_files_default():
    for json in listdir(org_list_default_dir):
        if json.endswith(".json"):
            yield open(join(org_list_default_dir, json))
160

# creating UCSCOrganism objects from json files

def get_organisms(json_files):
165     for json_file in json_files:
        for object in load(json_file):
            for child in object['children']:
                for grandchild in child['children']:
                    if grandchild['attr']['rel'] == 'genome':
170                         yield UCSCOrganism(db=grandchild['attr']['id'],
                                                title=grandchild['data']['title'])

```

3.2.2 Package rnaseqlyze.core

Modul rnaseqlyze.core

```

"""
Core functionality
"""

Modul rnaseqlyze.core.analysis
"""
Property getters and methods for Analysis instances
"""

5  import logging
log = logging.getLogger(__name__)

import os
from os.path import join, exists
10 import datetime
from urllib import quote

from sqlalchemy import ForeignKey
from sqlalchemy import Table, Column
15 from sqlalchemy import Boolean, Integer, String, Text, DateTime
from sqlalchemy.orm import relationship, backref, validates
from sqlalchemy.orm.properties import RelationshipProperty
from sqlalchemy.ext.declarative import declared_attr, declarative_base

20 import rnaseqlyze
from rnaseqlyze import galaxy
from rnaseqlyze import ucscbrowser
from rnaseqlyze.core import security

25 class Methods(object):
    """
    Here the various analysis configurations are handled
    as transparently as possible. The properties should be
    seasy to deal with so the worker.core code doesn't get too hairy.

30    .. note::

        Weather the input is an SRR identifier or a sra/fastq file is
        distinguished by checking "self.inputfile_name == None" in at least
35        - the Worker
        - the analysis.pt template
        - in this file

    .. note::

40        Weather the organism input is a NCBI RefSeq accession or a genbank file
        is distinguished by checking "self.genbankfile_name == None" in at least
        - the Worker
        - the analysis.pt template
45        - in this file
    """

    def __init__(self, **kwargs):

```

```

        super(Methods, self).__init__(**kwargs)
50     self.creation_date = datetime.datetime.utcnow()

    def create_data_dir(self):
        if not os.path.isdir(self.data_dir):
            os.makedirs(self.data_dir)

55     # org_db and hg_url (which depends upon org_db) are not set
    # as a db attribute, so old analyses where the organism was not
    # known at creation time automatically get the right url set if the
    # organism later on becomes available in the UCSC Browser

60     def get_hg_url(self, org_db):
        if not self.galaxy_hg_text:
            return
        hg_url = ucscbrowser.custom_track_url + \
65             ucscbrowser.custom_track_params.format(
                org_db=org_db, track_url=quote(self.hg_url))
        return hg_url

    def get_galaxy_id(self, name):
70     for ds in self.galaxy_datasets:
        if ds.name == name:
            return ds

class Properties(object):
75     """
    .. note::

        - `input` means `short reads data`
80     - `genbank` means `"genome" database nucleotide sequence`
    """

    # data uploaded or id specified ?
    # -----

85     @property
    def inputfile_uploaded(self):
        return self._inputfile_name and True

90     @property
    def genbankfile_uploaded(self):
        return self._genbankfile_name and True

    # directories
95     # -----

    @property
    def data_dir(self):
        return join(rnaseqlyze.analyses_path, str(self.id))

100    @property
    def input_data_dir(self):

```

```

        if self.inputfile_uploaded:
            return self.data_dir
105     else:
            return self.rnaseq_run.data_dir

    @property
    def genbank_data_dir(self):
110     if self.genbankfile_uploaded:
            return self.data_dir
        else:
            return join(rnaseqlyze.shared_data_path, self.org_accession)

115     # short reads files
    # -----

    @property
    def inputfile_name(self):
120     return self._inputfile_name \
            or self.rnaseq_run and self.rnaseq_run.srr + ".sra"

    @inputfile_name.setter
    def inputfile_name(self, value):
125     self._inputfile_name = value

    @property
    def inputfile_path(self):
        return join(self.input_data_dir, self.inputfile_name)

130     @property
    def inputfile_base_name(self):
        return self.inputfile_name.rsplit(".", 1)[0]

    @property
135     def inputfile_fq_name(self):
        return self.inputfile_base_name + ".fastq"

    @property
140     def inputfile_fq_path(self):
        return join(self.input_data_dir, self.inputfile_fq_name)

    @property
    def inputfile_header(self):
145     if not exists(self.inputfile_fq_path):
        return
        fq_file = open(self.inputfile_fq_path)
        lines = [fq_file.readline() for i in range(4)]
        log.info("Header: %s" % lines[0])
150     fq_file.close()
        return "".join(lines)

    # organism files
    # -----

155     @property

```

```

def genbankfile_name(self):
    return self._genbankfile_name \
        or self.org_accession and self.org_accession + ".gb"
160

@genbankfile_name.setter
def genbankfile_name(self, value):
    self._genbankfile_name = value

165
@property
def genbankfile_path(self):
    return join(self.genbank_data_dir, self.genbankfile_name)

@property
170
def genbankfile_base_name(self):
    return self.genbankfile_name.rsplit(".", 1)[0]

@property
def genbankfile_fa_name(self):
175
    return self.genbankfile_base_name + ".fa"

@property
def genbankfile_fa_path(self):
    return join(self.genbank_data_dir, self.genbankfile_fa_name)
180

@property
def xgenbankfile_name(self):
    return self.genbankfile_base_name + ".augmented.gb"

185
@property
def xgenbankfile_path(self):
    return join(self.data_dir, self.xgenbankfile_name)

# magic galaxy_xxx attributes
190 # -----

galaxy_stuff = "hg_text bam coverage hp_terms pr_operons".split()

for x in galaxy_stuff: exec """if True: # just to enable indentation ...
195     @declared_attr
    def galaxy_%s(self):
        return relationship("GalaxyDataset",
                            uselist=False, primaryjoin="%s")""" % (x,

200
                            "and_(GalaxyDataset.type == '%s', "
                            "Analysis.id == GalaxyDataset.analysis_id)" % x)

del x
@validates*(("galaxy_" + x for x in galaxy_stuff))
def _set_galaxy(self, attr, dataset):
205
    dataset.type=attr[7:]
    return dataset

# other things
# -----
210

```

```

@property
def hg_url(self):
    if not self.galaxy_hg_text:
        return
215     return "https://" + galaxy.hostname \
        + galaxy.dataset_display_url_template \
        .format(dataset=self.galaxy_hg_text.id)

@property
220 def data_dir_state(self):
    return hash(tuple((x,tuple(y),tuple(z))
        for x, y, z in os.walk(self.data_dir)))

@property
225 def stage_logs_state(self):
    return hash(tuple(self.stage_logs))

class Validators(object):
    @validates('org_accession')
230     def validate_org_accession(self, attr, acc):
        security.check_valid_filename(acc)
        return acc.upper()

    @validates('strandspecific', 'pairended')
235     def validate_boolean(self, attr, val):
        return val and True or False

    @validates('inputfile_name', 'genbankfile_name')
    def validate_x_file_name(self, attr, name):
240         if '\\\\' in name:
            name = name.rsplit('\\\\', 1)[1]
            security.check_valid_filename(name)
            if name.find('.') < 0:
                raise Exception("Please make sure your input file has a"
245                             " (meaningful) extension, like .fastq or .sra")
        return name

class Mixins(Methods, Properties, Validators):
    pass

```

Modul rnaseqlyze.core.entities

```

"""
SQLAlchemy Database Entities

A nice tutorial showing how everything works is `here
5     <http://docs.sqlalchemy.org/en/latest/orm/tutorial.html>`.
"""

from rnaseqlyze.core.orm import *

10 class Analysis(AnalysisMixins, Entity):
    # The order of superclasses matters!

```



```

# AnalysisMethods.__init__ calls Entity.__init__

15 """
    The central entity.
    Represents an analysis by a researcher.
    The whole rnaseqlyze project basically revolves around this entity.
    """

20
    id = Column(Integer, primary_key=True)

    org_gid = Column(Integer) # Organisms Genebank/Entrez gid
    org_accession = Column(String) # Organisms Genebank accession number

25
    _inputfile_name = Column("inputfile_name", String)
    inputfile_type = Column(String)
    _genbankfile_name = Column("genbankfile_name", String)

30
    strandspecific = Column(Boolean)
    pairedend = Column(Boolean)
    pairedlen = Column(Integer)

    owner = relationship("User", backref=backref("analyses"))
35
    owner_name = Column(String, ForeignKey('user.name'))

    creation_date = Column(DateTime)
    started = Column(Boolean)
    finished = Column(Boolean)
40
    stage = Column(String)
    error = Column(String)

    rnaseq_run = relationship("RNASeqRun", backref=backref("analyses"))
    rnaseq_run_srr = Column(String, ForeignKey('rnaseqrun.srr'))

45
    # ft_predictions = `backref` from FeaturePredictions
    # hg_tracks = `backref` from HgTrack
    # galaxy_datasets = `backref` from GalaxyDataset

50 class UploadSession(Entity):
    """
    Is created when somebody uploads a file
    """
    id = Column(Integer, primary_key=True)
55
    analysis_id = Column(Integer, ForeignKey(Analysis.id))
    analysis = relationship(Analysis, uselist=False)

    class User(Entity):
        """
60
        Constitutes a user of this service
        """
        name = Column(String, primary_key=True)
        # analyses = `backref` from Analysis

65
    def __init__(self, name):
        self.name = name

```

```

# SRA analogons

70 class RNASeqStudy(Entity): # stub
    """
    Constitutes an SRA "SRP" == SRA Study
    """
    srp = Column(String, primary_key=True)
75 # analyses = `backref` from Analysis
    # experiments = `backref` from RNASeqExperiment

class RNASeqExperiment(Entity): # stub
    """
80 Constitutes an SRA "SRX" == SRA Experiment
    """
    srx = Column(String, primary_key=True)
    srp_srp = Column(Integer, ForeignKey(RNASeqStudy.srp))
    srp = relationship(RNASeqStudy, backref=backref("experiments"))
85 # runs = `backref` from RNASeqRun

class RNASeqRun(RNASeqRunMixins, Entity):
    """
    Constitutes an SRA "SRR" == SRA Run
90 """
    srr = Column(String, primary_key=True)
    srx_srx = Column(Integer, ForeignKey('rnaseqexperiment.srx'))
    srx = relationship(RNASeqExperiment, backref=backref("runs"))

95 class UCSCOrganism(Entity):
    """
    Holds information about the mapping of UCSC browser "db" names to
    "gene id 'title'", and RefSeq Accessions.
    """
100 acc = Column(String, primary_key=True)
    db = Column(String, unique=True)
    title = Column(String, unique=True)

class GalaxyDataset(Entity):
105 """
    Holds a mapping from an analysis to a galaxy dataset id
    """
    id = Column(String, primary_key=True)
    analysis_id = Column(Integer, ForeignKey(Analysis.id), primary_key=True)
110 analysis = relationship(Analysis, backref=backref("galaxy_datasets"))
    type = Column(String)
    name = Column(String)

class StageLog(Entity):
115 """
    Holds the log output of one processing stage
    """
    # the primary key could be stage/analysis_id
    # but using an id automatically adds ordering
    # which comes handy, because how to order the stages otherwise ?
120 id = Column(Integer, primary_key=True)

```

```

    stage      = Column(String)
    analysis_id = Column(Integer, ForeignKey(Analysis.id))
    analysis    = relationship(Analysis, backref=backref("stage_logs"))
125    text      = Column(Text)

```

Modul rnaseqlyze.core.orm

```

"""
This module declares and imports everything needed to
define the database entity classes in :mod:`.entities`.
"""

5
from sqlalchemy import (
    ForeignKey,
    Table, Column,
    Boolean, Integer,
10    String, Text, DateTime
)
from sqlalchemy.orm import relationship, backref
from sqlalchemy.ext.declarative import declared_attr

15 from rnaseqlyze.core.analysis import Mixins as AnalysisMixins
from rnaseqlyze.core.srr import Mixins as RNASeqRunMixins

class _Entity(object):

20     @declared_attr
    def __tablename__(cls):
        return cls.__name__.lower()

    def __setattr__(self, name, value):

25         # raise an exception when setting
        # attributes that are not db columns
        if not (name.startswith('_') or hasattr(type(self), name)):
            raise Exception("%s' is not a declared attribute" % name)

30         super(_Entity, self).__setattr__(name, value)

from sqlalchemy.ext.declarative import declarative_base
Entity = declarative_base(cls=_Entity)
35 del declarative_base

```

Modul rnaseqlyze.core.security

```

"""
A collection of security related functions.

In case a check fails, an exception is raised, otherwise None is returned.
5 """

def check_valid_filename(name):
    """

```

```

10     Assert that the passed name doesn't contain
       any "funny" characters (e.g. ../../../../sensitive.txt)
       """

       max_len = 128
       assert len(name) < max_len, "Filename too long"

15
       import string
       assert set(name) < set(string.digits + string.letters + '._'), \
           "Only digits, letters, point and underscore allowed in filenames"

Modul rnaseqlyze.core.service

import logging
log = logging.getLogger(__name__)

import os
5 import urllib2

import rnaseqlyze
from rnaseqlyze.core import security
from rnaseqlyze.core.entities import Analysis, User, RNASeqRun, UploadSession
10 def get_upload_session(db_session):
    sess = UploadSession()
    db_session.add(sess)
    db_session.flush()
    return sess

15
def get_uploadfile(db_session, session, name, type):
    # This doesn't look right, but it works. The database needs to
    # be locked here to make sure that the first upload request that
    # comes in creates the analysis and the second uses the same analysis.
    # We need to lock the whole database and this seemingly useless statement
    # does just that. With SQLite. I have been asking on irc #sqlalchemy about
    # how to do it the right way, but I didn't get any useful reply. I have
    # checked the SQLAlchemy as well as the SQLite docs and tried various things
    # like DBSession.execute("BEGIN") and such things - nothing seems to work
    # - this is the only solution I have found.
    #
    # EDT: it could be as simple as increasing the sqlalchemy debug level, check
    # what sql statements are executed and then DBSession.execute() those
    #
    session.analysis = session.analysis
    #
    if not session.analysis:
        session.analysis = Analysis()
        db_session.add(session.analysis) # needed ?
    #
    # sets analysis.id
    db_session.flush()
    session.analysis.create_data_dir()

    assert type in ('inputfile', 'genbankfile')

40    typename = type + '_name'

```

```

# inputfile_name -> Short Reads in SRA or FASTQ format
# genabnkfile_name -> Organism genbank file
# if session.analysis.inputfile_name:
45 if getattr(session.analysis, type + '_uploaded'):
    # you land here if a user uploads
    # more than one file per type
    # this is not intended, BUT
    # these are the interwebs!
50     pass # FIXE: remove old

# session.analysis.inputfile_name = name
setattr(session.analysis, typename, name)

55 log.debug("creating upload file '%s' for analysis #%d" % (
    name, session.analysis.id))

# this would be the place to throw in a wrapper
# to track upload progress the old way, i.e.
60 # with server callbacks...

# inputfile_path is a @property
return open(getattr(session.analysis, type + '_path'), "w+b")

65 def get_analysis(db_session, attributes):

    # owner handling
    if 'owner' not in attributes:
        owner = db_session.query(User).get("anonymous")
70     if not owner:
        owner = User("anonymous")
        db_session.add(owner)
        attributes['owner'] = owner

75 # srr handling
rnaseq_run = None
if 'rnaseq_run' in attributes \
    and 'inputfile_name' not in attributes:
    log.debug("rnaseq_run: %s" % attributes['rnaseq_run'])
80     rnaseq_run = db_session.query(RNASEqRun).get(attributes['rnaseq_run'])
    if rnaseq_run:
        attributes['rnaseq_run'] = rnaseq_run
    else:
        try:
85             log.debug("creating new RNASEqRun")
            rnaseq_run = RNASEqRun(srr=attributes['rnaseq_run'])
            attributes['rnaseq_run'] = rnaseq_run
            rnaseq_run.create_directories()
            db_session.add(rnaseq_run)
90         except Exception, e:
            # The RNASEqRun constructor checks the SRRnnnnnnn argument
            # and raises an exception unless it passes the checks
            # e.g. if field was left blank/at default value
            # TODO: decide/document what to do
95             log.debug("failed: %r" % e)

```

```

        del attributes['rnaseq_run']

upload_session = db_session.query(UploadSession) \
    .get(attributes['upload_session'])
100 del attributes['upload_session']
    if not upload_session:
        raise Exception('this session has expired -'
            ' reload the "New Analysis" page to start a new one')

105 # the analysis exist already if the user uploaded something
    if upload_session.analysis:
        analysis = upload_session.analysis
        for attr, value in attributes.items():
            setattr(analysis, attr, value)
110
    else:
        # create db object
        log.debug("creating new analysis: %s" % attributes)
        analysis = Analysis(**attributes)
115 db_session.add(analysis)
        db_session.flush() # sets analysis.id
        analysis.create_data_dir()

    # allow no more uploads to this analysis
120 db_session.delete(upload_session)

    # if no input file has been uploaded
    if not analysis.inputfile_name:
        # an SRR identifier is needed
125         if rnaseq_run:
            analysis.rnaseq_run = rnaseq_run
        else:
            raise Exception("Please upload an input file or specify an SRR id")

130 return analysis

def start_analysis(analysis):
    url = "http://127.0.0.1:6543/analyses/%d"
    rq = RNASWorkerSTARTRequest(url % analysis.id)
135 opener = urllib2.build_opener(HTTPRNASWorkerHandler())
    rsp = opener.open(rq)
    body = rsp.read()
    rsp.close()

140 class RNASWorkerSTARTRequest(urllib2.Request):
    def get_method(self):
        return 'START'

class HTTPRNASWorkerHandler(urllib2.HTTPHandler):
145     def http_error(self, req, fp, code, msg, hdrs):
        raise WorkerException(fp.read())
    http_error_400 = http_error
    http_error_500 = http_error

```

```

150 class WorkerException(Exception):
    def __init__(self, exc_body):
        self.exc_body = exc_body
    def __repr__(self):
        return "WorkerException()"
155 def __str__(self):
    return self.exc_body

```

Modul rnaseqlyze.core.sra

Modul rnaseqlyze.core.srr

```

"""
Sequence Run Archive interaction
"""

5 import logging
log = logging.getLogger(__name__)

import os
from os import path
10 from urllib2 import urlopen
from shutil import copyfileobj

from sqlalchemy.orm import validates

15 import rnaseqlyze

url_template = "http://ftp-private.ncbi.nlm.nih.gov" \
    "/sra/sra-instant/reads/ByRun/sra/SRR/{srr:.6}/{srr}/{srr}.sra"
# e.g. "/sra/sra-instant/reads/ByRun/sra/SRR/SRR000/SRR000001/SRR000001.sra"
20
class Methods(object):
    def download(self):
        try:
            log.info("fetching " + self.srr)
            srr_url = url_template.format(srr=self.srr)
25            remote = urlopen(srr_url, timeout=60)
            local = open(self.sra_path, "w")
            copyfileobj(remote, local)
        except Exception, e:
            log.error("Error fetching SRR: %r" % e)
            os.unlink(self.sra_path)
            raise
        finally:
            # note:
35            # in case of an error, unlinking wil precede closing
            # -- no problem on unix
            if local:
                local.close()
            log.debug("done")
40
class Properties(object):

```

```

@property
def data_dir(self):
    return path.join(rnaseqlyze.shared_data_path, self.srr)
45
@property
def sra_path(self):
    return path.join(self.data_dir, self.sra_name)

50
@property
def sra_name(self):
    return self.srr + ".sra"

def create_directories(self):
55     if not os.path.isdir(self.data_dir):
        os.makedirs(self.data_dir)

class Validators(object):
    @validates('srr')
60     def check_srr(self, key, srr):
        import string
        assert len(srr) == 9
        assert srr[:3] == 'SRR'
        assert set(srr[3:]) < set(string.digits)
65         # ... what a powerful language python is! :-)
        # http://docs.python.org/library/stdtypes.html#set
        # http://docs.python.org/library/string.html#string-constants
        return srr.upper()

70 class Mixins(Methods, Properties, Validators):
    pass

```

3.2.3 Package rnaseqlyze.cli

Modul rnaseqlyze.cli

```

"""
Programs runnable from the command line.

For each module contained in this package, a wrapper script
5  called `rnas-<module name>` will be installed in `<prefix>/bin`.
"""

from .. import project_name
project_name += "-cli"

```

Modul rnaseqlyze.cli.apidoc

```

"""
RNA-Seqlyze ApiDoc Generator

Usage:
5  rnaseqlyze-apidoc -h|--help
  rnaseqlyze-apidoc [-s|--source] <path> ...

```


*Generates one <package>.rst sphinx apidoc source file,
in the current directory, for each package found in <path>.*

Options:

*-s --source Use `literalinclude` in addition to `automodule`.
When using this option, the generated output will be
optimized for processing with the sphinx latexpdf module that
generates a pdf document. Even without this option, when
using the html output module, the modules source code will
still be available in the generated Website, but not on the
same pages as the rest of the modules documentation
(controlled by the "html_show_sourcelink" option in
apidoc/conf.py).*

"""

```
import os, sys
from pkgutil import walk_packages
```

```
def main():
```

```
    import docopt
    opts = docopt.docopt(__doc__)
```

```
    global pkg_tpl
```

```
    global mod_tpl
```

```
    if opts['--source']:
```

```
        pkg_tpl, mod_tpl = pkg_src_tpl, mod_src_tpl
```

```
    #: implicit args to write():
```

```
    #: pkg.outfile, pkgpath, name, filename
```

```
    def write(tpl, **kwargs):
```

```
        pkg.outfile.write(tpl.format(
```

```
            name=name, path=pkgpath + os.sep + filename,
```

```
            equals="=" * len(name), dashes="-" * len(name), **kwargs))
```

```
    packages = {}
```

```
    Package = type('', (), {})
```

```
    # requirement/assumption:
```

```
    # parent packages will come before their children
```

```
    for loader, name, is_pkg in walk_packages(opts['<path>']):
```

```
        pkgname = name.rsplit('.', 1)[0]
```

```
        pkgpath = os.path.relpath(loader.path, ".")
```

```
        if is_pkg:
```

```
            # note: 'pkgname' is actually the
```

```
            # _parent_ package name in this case
```

```
            # associate non-root-packages with their parents
```

```
            if '.' in name:
```

```
                packages[pkgname].subpackages.append(name)
```

```
            # create & init a 'Package' object,
```

```
            # open <fully-qualified-package-name>.rst
```

```

        # and set filename to <package-name>/__init__.py
        pkg = Package()
        pkg.subpackages = []
        print "creating %s.rst" % name
        pkg.outfile = open(name + '.rst', 'w')
        filename = name.split('.')[0] + os.sep + '__init__.py'

        # add the package to the list
        # and write the packages .rst file heading
        packages[name] = pkg
        write(pkg_tpl)

    else:
        # skip modules that are not part of any package, like setup.py
        if pkgname not in packages:
            continue

        # find the containing package and set filename to <module>.py
        pkg = packages[pkgname]
        filename = name[len(pkgname)+1:] + '.py'

        # append the doc entry for this module to the package .rst file
        write(mod_tpl)

    if not opts['--source']:
        for pkg in packages.values():
            if pkg.subpackages:
                write(sub_pkg_tpl, names="\n\t".join(pkg.subpackages))

    pkg_tpl = """\
:mod: `{name}`
{equals}=====

.. automodule:: {name}

"""

    pkg_src_tpl = """\
:mod: `{name}`
{equals}=====

:mod: `{name}`
{dashes}-----

.. automodule:: {name}

Source Code:

.. literalinclude:: {path}

"""

    mod_tpl = """\
:mod: `{name}`

```

```

{dashes}-----

.. automodule:: {name}

120 """

mod_src_tpl = """\
:mod:`{name}`
{dashes}-----

125 .. automodule:: {name}

Source Code:

130 .. literalinclude:: {path}

"""

sub_pkg_tpl = """\
135 Subpackages
-----

.. toctree::
    :titlesonly:

140     {names}

"""

```

Modul rnaseqlyze.cli.galaxy_upload

```

"""
RNA-Seqlyze Galaxy-Upload

Usage:
5   rnas-galaxy-upload <local_file>
"""

import sys, os
from rnaseqlyze import galaxy

10 def main():

    if len(sys.argv) < 2 or sys.argv[1] in ('-h', '--help'):
        print __doc__
        return

15     print galaxy.upload(open(sys.argv[1]), os.path.basename(sys.argv[1]))

```

Modul rnaseqlyze.cli.gb2fasta

```

"""
RNA-Seqlyze gb2fasta

```

Convert a genbank file to fasta format

```

5
Usage:
    rnas-gb2fasta <input.gb> <output.fa>

If <input.gb> is '-', use 'sys.stdin', if <output.fa> is '-', use 'sys.stdout'.
10
"""
import sys
import Bio.SeqIO

def main():
15
    if len(sys.argv) < 2 or sys.argv[1] in ('-h', '--help'):
        print __doc__
        return

    inputfile = sys.argv[1] == '-' and sys.stdin or open(sys.argv[1])
    outputfile = sys.argv[2] == '-' and sys.stdout or open(sys.argv[2], "w")
    Bio.SeqIO.write(Bio.SeqIO.parse(inputfile, "genbank"), outputfile, "fasta")
20

```

Modul rnaseqlyze.cli.gb2ptt

```

"""
RNA-Seqlyze gb2ptt

Convert a genbank file to ptt (protein table) format
5
Usage:
    rnas-gb2ptt <input.gb> <output.ptt>

If <input.gb> is '-', use 'sys.stdin', if <output.ptt> is '-', use 'sys.stdout'.
10
"""
import sys, logging
from rnaseqlyze.gb2ptt import gb2ptt

def main():
15
    if len(sys.argv) < 2 or sys.argv[1] in ('-h', '--help'):
        print __doc__
        return

    inputfile = sys.argv[1] == '-' and sys.stdin or open(sys.argv[1])
    outputfile = sys.argv[2] == '-' and sys.stdout or open(sys.argv[2], "w")

    logging.basicConfig(level=logging.NOTSET) # logs to stderr

    gb2ptt(inputfile, outputfile)
25

```

Modul rnaseqlyze.cli.init

```

"""
RNA-Seqlyze Init

```

(Re-)initialize an rnaseqlyze 'workdir'.

Usage:

```
rnas-init <workdir>
rnas-init --recreatedb <workdir>
rnas-init --development <workdir>
rnas-init -h|--help
```

Options:

```
--recreatedb
    Remove and re-initialize the database if it exists.

--development
    Use the development versions of the config file templates.
```

Arguments:

```
<workdir>
    The filesystem path to the directory to be initialized.
    If the directory already exists, by default, existing
    files inside that directory are not overwritten.
```

.. important::

The `WORKDIR` variable in the "/etc/init.d/rnaseqlyze.sh" worker daemon startup script and the `workdir` variable in the "/var/www/./rna-seqlyze.wsgi" script must both be set to the directory specified here!

Documentation:

The 'workdir' holds

- configuration files (`*.ini`)
- the application database (`rnaseqlyze.db`)
- log files (`*.log`)
- shared data (`shared_data/`)
- individual analysis data (`analyses/`)

The rnas-init command

- creates the <workdir> if it does not already exist
- copies default configuration files 'rnaseqlyze.ini', 'web.ini' and 'worker.ini' into the <workdir> if they do not already exist.
- initializes the database that is configured in the 'rnaseqlyze.ini' config file if it doesn't already exist and --recreatedb is not given.

The database to be initialized is configured with "db_url =" in the "[rnaseqlyze]" section in 'rnaseqlyze.ini'. It is expected to be an sqlite database. If the command creates the sqlite database file, it changes it's unix access mode to (octal) 0664 and the group membership is changed to <group>. <group> can be configured in 'rnaseqlyze.ini'. If the command creates the <workdir>, it changes it's unix access mode to (octal) 0775 and the group membership is also changed to <group>. The command changes the unix access mode and group membership of all .log files inside the workdir to (octal) 0664 and <group>.

```

"""

60 import logging
log = logging.getLogger(__name__)

import os, sys, grp, shutil

65 import pkg_resources
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker

import rnaseqlyze
70 import rnaseqlyze.web
import rnaseqlyze.worker
from rnaseqlyze.core.entities import Entity

Session = sessionmaker()

75 def main():

    import docopt
    opts = docopt.docopt(__doc__)

80     workdir = os.path.abspath(opts['<workdir>'])

    # create the workdir if it does not exist
    wd_created = False
85     if not os.path.isdir(workdir):
        if os.path.exists(workdir):
            log.error("not a directory: '%s'" % workdir)
            sys.exit(1)
        log.info("creating workdir '%s'" % workdir)
90     os.makedirs(workdir)
    wd_created = True

    # create each config file that does not exist
    for pkg in rnaseqlyze, rnaseqlyze.web, rnaseqlyze.worker:
95
        # determine the destination file name
        conf_name = pkg.__name__.split('.')[1] + ".ini"
        conf_path = os.path.join(workdir, conf_name)
        if os.path.exists(conf_path):
100             continue

        # determine the source file name
        if pkg.__name__ == "rnaseqlyze":
            # for the core package there is currently
105             # only one config file template
            ini = 'rnaseqlyze.ini'
        else:
            # for the non-core packages, take the desired version
            ini = opts['--development'] and "development.ini" or "production.ini"
110     # get the file as a resource stream, which works even
    # if the distribution is installed as a zipped .egg

```

```

    req = pkg_resources.Requirement.parse(pkg.project_name)
    res = pkg_resources.resource_stream(req, ini)
    log.info("creating config file '%s'" % conf_name)
115     shutil.copyfileobj(res, open(conf_path, "w"))

    # init rnaseqlyze configuration -- creates all .log files
    rnaseqlyze.configure(workdir)

120     # set proper permissions on the log files
    for name in os.listdir(workdir):
        if name.endswith('.log'):
            path = os.path.join(workdir, name)
            log.info("adjusting permissions on '%s'" % name)
125             os.chmod(path, 0664)
            os.chown(path, -1, grp.getgrnam(rnaseqlyze.group).gr_gid)

    # delayed because 'rnaseqlyze.group' was
    # not known before calling rnaseqlyze.configure() above
130     if wd_created:
        log.info("adjusting permissions on '%s'" % workdir)
        # change permission bits
        os.chmod(workdir, 0775)
        # change group membership
135         os.chown(workdir, -1, grp.getgrnam(rnaseqlyze.group).gr_gid)

    # get the database file path
    db_path = rnaseqlyze.db_url.split(":", 1)[1]

140     # remove the database file
    # if it exists and --recreatedb is given
    if os.path.exists(db_path) and opts['--recreatedb']:
        log.info("removing existing database file '%s'" %
145                    db_path.split('/')[-1])
        os.unlink(db_path)

    # create the database if it doesn't exist
    if not os.path.exists(db_path):

150         log.info("recreating database '%s'" % rnaseqlyze.db_url)

        # create sqlalchemy db engine
        engine = create_engine(rnaseqlyze.db_url)

155         # create the file and initialize the schema
        with engine.begin() as conn:
            Entity.metadata.create_all(conn)

        log.info("adjusting permissions on database file")

160         # change permission bits
        os.chmod(db_path, 0664)
        # change group membership
        os.chown(db_path, -1, grp.getgrnam(rnaseqlyze.group).gr_gid)
165

```

```

        log.info("initializing organism cache")

        # initialize UCSC Browser list of organisms
        from rnaseqlyze import org_cache
170     with engine.begin() as conn:
            session = Session(bind=conn)
            org_cache.refresh(session)
            session.commit()

175     log.info("workdir initialized")

```

Modul rnaseqlyze.cli.transterm

```

"""
RNA-Seqlyze transterm

Calls the transterm program
5 with an additional "-p <path to>/expterm.dat" argument.

Usage:
    rnas-transterm [--] <transterm arguments> ...
    rnas-transterm -h|--help
10 """

import sys
from rnaseqlyze.transterm import run

def main():
15
    if len(sys.argv) > 1 and sys.argv[1] == '--':
        sys.argv.pop(1)
    elif len(sys.argv) < 2 or sys.argv[1] in ('-h', '--help'):
        print __doc__
20    return

    run(sys.argv[1:])

```

Modul rnaseqlyze.cli.xmltool

```

"""
RNA-Seqlyze xmltool

XML version of of `python -m json.tool`.
5 Takes an xml file or stream as input and pretty-prints it.

Usage:
    rnas-xmltool <input.xml> <output.xml>
10

If <input.xml> is '-', use 'sys.stdin, if <output.xml> is '-', use 'sys.stdout'.
"""

import sys
from lxml import etree
15

```



```

def main():

    if len(sys.argv) < 2 or sys.argv[1] in ('-h', '--help'):
        print __doc__
20         return

    inputfile = sys.argv[1] == '-' and sys.stdin or open(sys.argv[1])
    outputfile = sys.argv[2] == '-' and sys.stdout or open(sys.argv[2], "w")

25     tree = etree.parse(inputfile,
                        etree.XMLParser(remove_blank_text=True))
    print >> outputfile, etree.tostring(tree.getroot(), pretty_print=True)

```

3.2.4 Package rnaseqlyze.web

Modul rnaseqlyze.web

```

"""
**pyramid.web** is a Pyramid Web Framework Application.

To learn more about the applications architecture, head over to the wonderful
5 world of the pyramid web framework at http://www.pylonsproject.org/.

This application has been created using the ``pcreate`` command with the ``-s
alchemy`` option to create and sqlalchemy scaffold. There is plenty of `very
good documentation <http://docs.pylonsproject.org/projects/pyramid/en/latest/\narr/project.html#scaffolds-included-with-pyramid>`_ available on how to do it.
10

In case you have trouble with anything pyramid-related, use the `source code on
github <https://github.com/Pylons/pyramid>`_ or ask 'mcdonc' on freenode irc
`#pyramid <http://webchat.freenode.net/?channels=#pyramid>`_.
15 """

import logging
log = logging.getLogger(__name__)

20 from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker, scoped_session
from pyramid.config import Configurator
#: the zope transaction extension
from zope.sqlalchemy import ZopeTransactionExtension

25 import rnaseqlyze
from rnaseqlyze.web.jsonx import jsonx
project_name = rnaseqlyze.project_name + "-web"

30 #: a session managed by
#: ZopeTransactionExtension
#:
#: - http://stackoverflow.com/a/6044925
#: - pyramid_tm (transaction manager) is configured
35 DBSession = scoped_session(sessionmaker(extension=ZopeTransactionExtension()))

```

```

#: an unmanaged session
#:
#: used by :meth:`rnaseqlyze.web.views.post`
40 #: because the session needs to be committed early there
DBSession_unmanaged = scoped_session(sessionmaker())

def main(global_config, **settings):
    """
45    Create and return a Pyramid WSGI application.
    """
    log.debug("rnaseqlyze.web version %s : main()" % rnaseqlyze.__version__)

    # make sure to be able to delete files created by webapp
    # as user/group www-data/www-data from the command line
    # (as user/group johndoe/www-data)
    import os
    os.umask(0002)

55    engine = create_engine(rnaseqlyze.db_url)
    DBSession.configure(bind=engine)
    DBSession_unmanaged.configure(bind=engine)

    config = Configurator(settings=settings)
60    config.add_renderer('jsonx', jsonx)

    config.scan()

65    config.add_route('home', '/')
    config.add_route('upload', '/upload')
    config.add_route('analyses', '/analyses')
    config.add_route('analysis', '/analyses/{id}')
    config.add_route('analysis_files', '/analyses/{id}/files*subpath')
70    config.add_route('analysis_rest', '/rest/analyses/{id}')
    config.add_route('analysis_logs_rest', '/rest/analyses/{id}/logs')
    config.add_route('analysis_files_rest', '/rest/analyses/{id}/files')

75    config.add_route('organisms_rest', '/rest/organisms')

    for path in 'less', 'css', 'img', 'js':
        config.add_static_view(path, path)

80    return config.make_wsgi_app()

from pyramid.events import subscriber
from pyramid.events import BeforeRender
from pyramid.renderers import get_renderer

85 @subscriber(BeforeRender)
def before_render(event):
    """
    This function is called by Pyramid after the view callable has returned
    and before the renderer (json / chameleon) is called. We inject some
90

```

```

convenience functions and objects that are used in the `zope page
templates <http://pagetemplates.org/docs/latest/reference.html>`_
(.pt files) which `the chameleon template engine
<http://pagetemplates.org/>`_ then renders.
"""
95

base = get_renderer('templates/base.pt').implementation()

rq = event['request']
100 path = lambda sub: rq.route_path('home') + sub
relpath = lambda sub: rq.current_route_path() + '/' + sub

event.update({
    'base': base,
105    'path': path,
    'relpath': relpath,
    'version': rnaseqlyze.__version__,
    'debug': log.getEffectiveLevel() <= logging.DEBUG,
})

```

Modul rnaseqlyze.web.errors

```

"""
Pyramid Application Custom Error Views
"""

5 import logging
log = logging.getLogger(__name__)

from string import Template

10 from pyramid.view import view_config
from pyramid.response import Response, FileResponse
from pyramid.httpexceptions import (
    HTTPFound, HTTPError, HTTPServiceUnavailable, HTTPInternalServerError
)

15 import transaction
from sqlalchemy.exc import DBAPIError

import rnaseqlyze
20 from rnaseqlyze.web import DBSession, DBSession_unmanaged
from rnaseqlyze.core import service
from rnaseqlyze.core.entities import Analysis

@view_config(context=Exception)
25 def error(request):
    """
    **Exception view**

    This is a catch-all view that serves up any errors
    that have occurred while processing the a request.
30

    The view just creates and returns a custom error response object.

```

```

    """
    return HTTPRNASeqError(request.exc_info)
35
class HTTPRNASeqError(HTTPError):
    """
    Custom HTTP Error class.

40    This is a custom HTTP error class that extends
    :class:`pyramid.httpexceptions.HTTPError`, which extends
    :class:`pyramid.httpexceptions.WSGIHTTPException`. Have a look at the
    `source code <http://git.io/CqrjOg#L157>`_ to see how it works.

45    It's ``code`` is 500, which generally means "Internal Server Error". If the
    application is in debugging mode -- i.e. the log level is DEBUG or less, a
    stack trace is added to the generated page as well as the log file.
    Otherwise, an informational message is displayed and only one line,
    containing the type of the error is logged.
50    """
    code = 500
    title = "RNA-Seqlyze Web Application Error"
    explanation = "An Exception was raised in rnaseqlyze.web"
    html_template_obj = Template(Template('\n'.join(map(lambda s: s[8:], """\
55        <html>
        <head>
        <title>${title}</title>
        </head>
        <body style="margin: 20px;">
60        <h1>${title}</h1>
        ${body}
        </body>
        </html>
        """.split('\n')))).safe_substitute(title=title))
65    def __init__(self, exc_info):
        e = exc_info[1]
        log.error(repr(e))
        body_template = "<b>${explanation}</b>\n<hr/>\n"
        cls = e.__class__.__name__
70        if not e.args:
            self.explanation = "%s" % cls
        else:
            self.explanation = "%s: %s" % (cls, e.args[0])

75        if log.getEffectiveLevel() > logging.DEBUG:      # no debug
            detail = production_error_msg % \
                rnaseqlyze.admin_email
            body_template += "${detail}"
        else:                                              # debug
80            detail = ''
            if isinstance(e, DBAPIError):
                detail += dberror_msg
            import traceback
            detail += '%s\n\nStack trace:\n' % e
85            detail += ''.join(traceback.format_tb(exc_info[2]))

```

```

        log.debug(detail)
        body_template += "<pre>\n${detail}</pre>"

90         HTTPError.__init__(self, detail, body_template=body_template)

    dberror_msg = """\
    This is a database related error.

95    If it is not yet initialized or the schema has changed,
    just run the "rnas-dbinit" script to (re-)initialize it.

    Afterwards, restart the Pyramid application, i.e. send a
    SIG_INT to the apache mod_wsgi daemon processes, and try again.
100    """

    production_error_msg = """\
    If you think that this is a bug, please contact the application administrator,
    %s, and inform him/her of the time the error occurred, and anything you might
105    have done that may have caused the error.
    Thank You!
    """

    # 'You' is intentionally capitalized! :-) Rule 84: http://goo.gl/BLBuX

```

Modul rnaseqlyze.web.jsonx

```

    """
    Pyramid JSON renderer that serializes arbitrary objects

    Copies the object's __dict__, looks up all attrs
5    in all base classes's __dict__'s on the object
    and then strips any unknown attribute types.
    """

    #import logging
10    #log = logging.getLogger(__name__)

    import json

    #: a custom json renderer
15    jsonx = lambda info: render_json

    def render_json(value, system):
        """
        custom json renderer implementation

20        based on http://git.io/a6BFGQ#L169
        """

        request = system.get('request')
25        if request is not None:
            response = request.response
            response.content_type = 'application/json'
            return json.dumps(value, default=render_object, indent=4)

```

```

30 def render_object(obj):
    """
    "default" function for json.dumps()
    """
    attrs = dict((attr, getattr(obj, attr))
35         for base in obj.__class__.__mro__
         for attr in base.__dict__
         if attr[0] != '_')

    attrs.update(obj.__dict__)
40 # log.debug(attrs)
    return dict(filter(filter_attributes, attrs.iteritems()))

none_type = type(None)
def filter_attributes(kv):
45     """
    Helper function for render_object
    """

    if kv[0][0] != '_' and \
        type(kv[1]) in (none_type, bool, int, long, float, str, unicode, list):
50         return True
    return False

```

Modul rnaseqlyze.web.rest

```

    """
    Pyramid REST Views
    """
    import logging
5 log = logging.getLogger(__name__)

    import os

    from pyramid.view import view_config
10
    import rnaseqlyze
    from rnaseqlyze.web import DBSession, DBSession_unmanaged
    from rnaseqlyze.core import service
    from rnaseqlyze.core.entities import Analysis, StageLog, UCSCOrganism
15
    @view_config(route_name='analysis_rest', renderer='jsonx')
    def display(request):
        """
        **REST Analysis View**
20         """

        analysis = DBSession.query(Analysis).get(int(request.matchdict["id"]))
        first = DBSession.query(UCSCOrganism) \
            .filter(UCSCOrganism.acc.like(
                analysis.org_accession + '%')) \
            .first()
25
        org_db = first and first.db
        analysis.__dict__.update({
            'org_db': org_db,
            'hg_url': analysis.get_hg_url(org_db)})

```

```

    return analysis
30
@view_config(route_name='analysis_logs_rest', renderer='jsonx')
def analysis_stage_logs(request):
    """
    ***REST Stage Logs View***
    """
35
    criterion = StageLog.analysis_id == int(request.matchdict["id"])
    logs = DBSession.query(StageLog).filter(criterion).all()
    return sorted(logs, key=lambda log: log.id)

40 @view_config(route_name='analysis_files_rest', renderer='jsonx')
def analysis_files(request):
    """
    **REST Files View**
    """
45
    This view provides a (minimalistic, only GET is
    implemented) REST interface to '/analysis/{id}/files'.
    """
    files = []
    analysis = DBSession.query(Analysis).get(int(request.matchdict["id"]))
    os.chdir(analysis.data_dir)
50
    for dirpath, dirnames, filenames in os.walk("."):
        dir = dirpath[2:]
        for fn in filenames:
            files.append({'path': os.path.join(dir, fn)})
55
    return files

@view_config(route_name='organisms_rest', renderer='jsonx')
def organisms(request):
    """
60
    ***REST Organisms View***

    Displays the list of organism titles
    along with their UCSC db and NCBI RefSeq accession identifiers
    """
65
    return DBSession.query(UCSCOrganism).all()

```

Modul rnaseqlyze.web.upload

```

    """
    Pyramid Application Upload View

    This module handles the upload of analysis files.
5
    The upload interface consists of `plupload`
    (from http://www.plupload.com/) on the client
    and this hack on the server side. Combining the
    two and creating a working solution was not trivial.
10
    Documentation and inspiration to create this
    was, amongst others, taken from the following documents:

    - http://www.plupload.com/documentation.php

```

```

15  - http://hg.python.org/cpython/file/2.7/Lib/cgi.py#l353
    - https://raw.githubusercontent.com/moriecode/plupload/master/examples/upload.php
    - https://github.com/hcwebdev/plupload/blob/master/examples/server.py
    - https://github.com/Pylons/webob/blob/master/webob/request.py#L102
    - https://hg.gawel.org/gp.fileupload/file/default/gp/fileupload/storage.py#l97
20  """

    import logging
    log = logging.getLogger(__name__)

25  import transaction
    from pyramid.view import view_config

    from rnaseqlyze.web import DBSession
    from rnaseqlyze.core import service
30  from rnaseqlyze.core.entities import UploadSession

    @view_config(route_name='upload', request_method='POST', renderer="json")
    def upload(request):
        log.debug("upload(): content-type '%s'" % request.content_type)
35        fs = FieldStoragx(fp=request.environ['wsgi.input'], environ=request.environ)
        return dict(jsonrpc="2.0", result=None, id=None)

    import cgi
    class FieldStoragx(cgi.FieldStorage):
40        def __init__(self, fp=None, headers=None, outerboundary="",
                      environ=None, keep_blank_values=0, strict_parsing=0):
            self.environ = environ
            cgi.FieldStorage.__init__(self, fp, headers, outerboundary,
                                      environ, keep_blank_values, strict_parsing)

45        if self.filename:
            return
        assert len(self.value) < 1000
        if self.name == 'session':
            environ['rnaseqlyse.upload_session'] = \
50                DBSession.query(UploadSession).get(int(self.value))
        elif self.name in ('name', 'type'):
            environ['rnaseqlyse.upload_' + self.name] = self.value
        else:
            return

55        log.debug("FieldStoragx(%s -> %s)" % (self.name, self.value))

    def make_file(self, binary=None):

        assert self.filename
60        log.debug("FieldStoragx.make_file(%s)" % self.filename)

        args = {}
        for kw in 'session', 'name', 'type':
            args[kw] = self.environ['rnaseqlyse.upload_' + kw]

65        fd = service.get_uploadfile(DBSession, **args)
        # commit the (managed) session early here, so later
        # requests can re-use the Analysis object that the

```



```

70     # first one has implicitly created by calling
    # service.get_uploadfile
    import transaction
    transaction.commit()

    return fd

Modul rnaseqlyze.web.views
"""
Pyramid Application User Views
"""

5  import logging
    log = logging.getLogger(__name__)

    import re
    from os.path import join

10     from pyramid.view import view_config
    from pyramid.response import FileResponse
    from pyramid.httpexceptions import HTTPFound

15     import transaction
    from sqlalchemy.exc import DBAPIError

    import rnaseqlyze
    from rnaseqlyze import galaxy
20     from rnaseqlyze.web import DBSession, DBSession_unmanaged
    from rnaseqlyze.core import service
    from rnaseqlyze.core.entities import Analysis, UCSCOrganism

    autocomplete_re = re.compile(r"^[^()]+\([^~/\+/(~)]+\)\.*$")

25     @view_config(route_name='home', renderer='templates/home.pt')
    def home(request):
        """
        **Home Page**

        This is the main entry point to the application. I.e. the landing page,
        the page that users see first.
        """
        return {}

35     @view_config(route_name='analyses', renderer='templates/create.pt')
    def create(request):
        """
        **Create Page**

        This page is shown when the "New Analysis" button is clicked.
        """
        sess = service.get_upload_session(DBSession)
        return { 'upload_session': sess.id }

45

```

```

@view_config(route_name='analysis', renderer='templates/analysis.pt')
def display(request):
    """
    **Analysis Page**

    This page is displayed after the the anaysis has been created.
    When the user clicks "Submit" on the 'create' page, after
    the files are uploaded and the form information is submitted
    to the :func:`~post` view, the browser is redirected here.

    The page can also be viewed any time later on, no matter
    weather the analysis has already been completed or not.

    In case it is not yet completed, the page is constantly
    updated via XMLHTTPRequests to reflect the current status.
    """

    id = int(request.matchdict["id"])
    return {
        'analysis': DBSession.query(Analysis).get(id),
        'galaxy_history_url': galaxy.default_history_url,
    }

@view_config(route_name='analysis_files')
def analysis_files(request):
    """
    **Files View**

    This view serves up the files associated with
    an analysis on 'http://<rnaseqlyze>/analysis/{id}/files'.
    """

    return FileResponse(join(rnaseqlyze.analyses_path,
                             request.matchdict['id'], *request.subpath))

import mimetypes
#mimetypes.add_type("text/plain", ".")
#mimetypes.add_type("text/plain", ".")
#mimetypes.add_type("text/plain", ".")
#mimetypes.add_type("text/plain", ".")
mimetypes.add_type("text/plain", ".gb")
mimetypes.add_type("text/plain", ".log")
mimetypes.add_type("text/plain", ".log0")
mimetypes.add_type("text/plain", ".info")
# FileResponse automatically sets the Content-Type header based on this

@view_config(route_name='analyses', request_method='POST')
def post(request):
    """
    **Create-Form Action**

    This view just redirects the client to the created analysis page.
    Before it is actually called, the files to be analyzed, are uploaded
    using the :func:`~upload.upload` view callable.
    """

```

```

100     # for documentation on the documentation reference syntax, see
    # http://sphinx.pocoo.org/domains.html#cross-referencing-python-objects

    # TODO: csrf security checks
    #     see "shootout" pyramid demo app
105
    # note:
    # when using the "DBSession" (managed), the
    # try:/except: rollback construct is not needed
    # because the session is automatically rolled back
110    # otoh, if the _unmanaged session is used, it _has_ to
    # be manually committed or rolled back if objects are modified

    if 'org_accession' in request.POST:
        request.POST['org_accession'] = \
115            autocomplete_re.sub(r"\1", request.POST['org_accession'])
    try:
        analysis = service.get_analysis(
            DBSession_unmanaged, attributes=request.POST)

120        # the analysis must exist in the database
        # so the worker can find it and start working
        DBSession_unmanaged.commit()

        service.start_analysis(analysis)
125        log.debug("started analysis #%d by '%s'" % (
            analysis.id, analysis.owner.name))

        return HTTPFound(request.route_path('analysis', id=analysis.id))
    except:
130        log.info("abort")
        DBSession_unmanaged.rollback()
        log.debug("rollback complete")
        raise

```

Modul rnaseqlyze.web.wsgi

```

"""
RNA-Seqlyze WSGI Application

Provides the get_app(workdir) function,
5  which returns a wsgi application callable.
"""

def get_app(workdir):
    """
10    Basically returns wrapper around paster.get_app
    that strips the ".wsgi" extension from SCRIPT_NAME
    """

    # configure the core package
15    import rnaseqlyze
    rnaseqlyze.configure(workdir)

```

```

# default configuraion file name
import os.path
20 web_ini = os.path.join(workdir, 'web.ini')

# configure logging
import logging.config
logging.config.fileConfig(web_ini, dict(here=workdir))
25

# create the pyramid wsgi app
import pyramid.paster
pyramid_app = pyramid.paster.get_app(web_ini, 'main')

# return a wrapper that adjusts SCRIPT_NAME
30 def app(environ, start_request):
    environ['SCRIPT_NAME'] = \
        environ['SCRIPT_NAME'][:-5]
    return pyramid_app(environ, start_request)
35

return app

```

3.2.5 Package rnaseqlyze.worker

Modul rnaseqlyze.worker

```

"""
**pyramid.worker** is a Pyramid Web Framework Application.

The framework is used here to keep things simple. Even though not many of the
5 frameworks features are used, building this "-worker" part of the project as a
Pyramid Web Framework Application, just like the "-web" part, hopefully makes
it easy to understand for anybody already understanding the "-web" part.

The key features from the Pyramid Web Framework used here, are
10
1) The "pserve" command, which makes running the application as a unix daemon
process very simple. It's direct use has actually been deprecated during
the development and a custom command, "rnas-worker", has been created,
which uses the same python functions and modules like "pserve".
15
2) The pyramid.config.Configurator class, that is used to define the
applications "routes" and "view callables". These "views" provide the
applications interface. They are served on a tcp port bound to localhost
(127.0.0.1) and are therefore only available to processes running on the
20 same host.

The "-worker" applications interface has "HTTP-like" semantics.

The following commands are accepted:
25
- ``GET /analyses/{id}``: Show the current status.
- ``START /analyses/{id}``: Start processing an analysis.

Only for development purposes, one additional command exists:

```

```

30     - ``RESTART /analyses/{id}``: Restart an analysis
                                   that has already been started.

The commands are executed by the "-web" part of the application by subclassing
35 HTTPRequest and overridding the get_method() function. They can also be executed
from the command line however, using the popular "curl" binary with the "-X"
option, e.g.

    - ``curl -X GET localhost:/analyses/3``
40     - ``curl -X START localhost:/analyses/3``
    - ``curl -X RESTART localhost:/analyses/3``

"""

45 import logging
log = logging.getLogger(__name__)

from pyramid.view import view_config
from pyramid.view import view_defaults
50 from pyramid.config import Configurator
from pyramid.response import Response
from pyramid.httpexceptions import (
    HTTPError,
    HTTPBadRequest,
55     HTTPInternalServerError,
)

from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker, scoped_session
60 from zope.sqlalchemy import ZopeTransactionExtension

import rnaseqlyze
project_name = rnaseqlyze.project_name + "-worker"

65 from rnaseqlyze.core.entities import Analysis
from rnaseqlyze.worker.core import (
    Manager,
    ManagerBusyException,
    AnalysisAlreadyStartedException,
70 )

DBSession = scoped_session(sessionmaker(extension=ZopeTransactionExtension()))

def main(global_config, **settings):
75     """
    Return a Pyramid(!) WSGI application.
    """

    # make sure to be able to delete files created by webapp
    # as user/group www-data/www-data from the command line
    # (as user/group johndoe/www-data)
80     import os
    os.umask(0002)

```

```

85     engine = create_engine(rnaseqlyze.db_url)
        DBSession.configure(bind=engine)

        Waitress.manager = Manager()

90     config = Configurator(settings=settings)
        config.add_route('analyses', '/analyses/{id}')
        config.scan()
        return config.make_wsgi_app()

95 @view_defaults(route_name='analyses', renderer='string')
    class Waitress(object):

        def __init__(self, request):
100             id = int(request.matchdict['id'])
                self.analysis = DBSession.query(Analysis).get(id)

        @view_config(request_method='GET')
        def status(self):
105             import pprint
                return pprint.pformat({
                    'context': self, # Waitress
                    'manager': self.manager, # Manager
                    'analysis': self.analysis, # Analysis
110             })

        @view_config(request_method='START')
        def start(self):
                self.manager.analysis_requested(self.analysis)
115             return "started analysis #%d" % self.analysis.id

        @view_config(request_method='RESTART')
        def restart(self):
                self.manager.analysis_requested(self.analysis, True)
120             return "restarted analysis #%d" % self.analysis.id

        @view_config(context=Exception)
        def error_view(error, request):
125             errdict = {
                AnalysisAlreadyStartedException: HTTPBadRequest,
                ManagerBusyException: HTTPInternalServerError,
            }
            if isinstance(error, HTTPError):
130                 return error
            elif type(error) in errdict:
                return errdict[type(error)](error)
            else:
                return HTTPInternalServerError(error)

135
        # monkey-patch some HTTPException classes to get simpler error messages

```

```

from pyramid.response import Response
140 def _WHE_init(self, arg=None):
    Exception.__init__(self, arg)
    if isinstance(arg, Exception):
        if False: # production
            e, t = arg, type(arg)
            arg = "%s %s" % (t.__name__, e.args)
145         else: # debug
            import traceback
            arg = traceback.format_exc(999)
    Response.__init__(self,
150         '%s %s\n\n%s' % (self.code, self.title, arg),
        content_type='text/plain', status='%s %s' % (self.code, self.title))

from pyramid.httpexceptions import WSGIHTTPException
WSGIHTTPException.__init__ = _WHE_init
155 del WSGIHTTPException.__call__
del WSGIHTTPException.prepare

```

Modul rnaseqlyze.worker.core

```

"""
RNA-Seqlyze Worker Daemon Core

Worker parent class with basic infrastructure to run the
5 various analysis steps defined in :class:`~.WorkerStages`.
"""

import logging
log = logging.getLogger(__name__)
10 root_logger = logging.getLogger()

from threading import Thread
from logging import Formatter
from logging import StreamHandler
15 from StringIO import StringIO
from contextlib import contextmanager

from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker
20
import rnaseqlyze
from rnaseqlyze import efetch
from rnaseqlyze.core.entities import Analysis, StageLog
from rnaseqlyze.worker.stages import WorkerStages
25
DBSession = sessionmaker()

log_format = "%(levelname)-5.5s [%s] %(message)s"

30 class Manager(object):
    def __init__(self):
        self.worker = Thread()

```

```

def analysis_requested(self, analysis, re=False):
35     if analysis.started and not re:
        raise AnalysisAlreadyStartedException
    if self.worker.is_alive():
        raise ManagerBusyException
    self.worker = Worker(analysis)
40     self.worker.start()

class StageLogStream(object):
    def __init__(self, analysis, stage, session):
        self.stage_log = StageLog(analysis=analysis, stage=stage, text="")
45     self.session = session
        session.add(self.stage_log)
        session.commit()
    def write(self, data):
        self.stage_log.text += data
50     self.session.commit()

class AnalysisAlreadyStartedException(Exception):
    pass

55 class ManagerBusyException(Exception):
    pass

class Worker(Thread, WorkerStages):
    """
60     The Worker
    """

    def __init__(self, analysis):
        Thread.__init__(self)
65     self.analysis_id = analysis.id

    def _thread_init(self):
        from os import path
        self.session = DBSession(bind=create_engine(rnaseqlyze.db_url))
70     self.analysis = self.session.query(Analysis).get(self.analysis_id)

        self.logfile = open(path.join(
            self.analysis.data_dir, "rna-seqlyze-worker.log"), "w")
        self.log_handler = StreamHandler(self.logfile)
75     self.log_handler.setFormatter(Formatter(log_format))
        root_logger.addHandler(self.log_handler)
        log.info("starting work on analysis #%d" % self.analysis_id)
        self.analysis.finished = False
        self.analysis.started = True
80     self.analysis.error = None
        self.session.commit()

    @contextmanager
    def _stage_log_manager(self, stage):
85     handler = StreamHandler(
        StageLogStream(self.analysis, stage, self.session))
        root_logger.addHandler(handler)

```



```

        yield
        root_logger.removeHandler(handler)
90
    def run(self):

        # TODO: invent a way to avoid calling stages that won't do anything
        #       maybe @stages -> @stages(condition) something ...
95

        self._thread_init()
        try:
            for stage in self.stages:
                if not stage.should_run(self):
                    continue
100                log.info("=== %s ===" % stage.func_name)
                with self._stage_log_manager(stage.func_name):
                    self.analysis.stage = stage.func_name
                    self.session.commit()
                    stage(self)
105        except Exception, e:
            self.analysis.error = repr(e)
            raise
        finally:
            if self.analysis.error:
                log.error(self.analysis.error)
            else:
                log.info("analysis finished")

115        self.analysis.finished = True
        self.session.commit()
        root_logger.removeHandler(self.log_handler)
        self.logfile.close()

```

Modul rnaseqlyze.worker.daemon

```

"""
RNA-Seqlyze Worker

Start, stop or restart the worker daemon
5 or run it in the foreground, in development mode.

Usage:
    rnas-worker <workdir> (start|stop|restart)
    rnas-worker <workdir> --development
10    rnas-worker -h|--help

Arguments:

    <workdir> The path to the workers 'workdir'.
15    The 'workdir' is where the configuration, the
    application database and all analysis data are stored.

    start|stop|restart

20    If one of those arguments is given, the daemon is

```

```

run in the background. It will write it's PID to the
file <workdir>/worker-daemon.pid and its output will be logged
to <workdir>/worker-daemon.log. This is not the "log file"
however. The "log file" path can be configured
in <workdir>/worker.ini.
25

--development

If this argument is present, the worker daemon is run in
development mode, which means that it will no fork to the
background. If any source files (.py) are changed when the
daemon is running in development mode, it will be
automatically restarted.
30

"""
35

from os.path import abspath, join

from paste.script import serve
import docopt
40

import rnaseqlyze

def main():
45     opts = docopt.docopt(__doc__)

    for command in "start|stop|restart".split('|'):
        if opts[command]:
            mode = "production"
            args = [command, "--daemon"]
50             break
        else:
            mode = "development"
            args = ["--reload"]
55

    workdir = abspath(opts['<workdir>'])
    rnaseqlyze.configure(workdir)

    if mode == 'production':
60         args.extend([
            # "--user=" + rnaseqlyze.worker_user,
            # "--group=" + rnaseqlyze.group,
            "--log-file=" + join(workdir, 'worker-daemon.log'),
            "--pid-file=" + join(workdir, 'worker-daemon.pid'),
65         ])

    conf_file = join(workdir, 'worker.ini')
    serve.ServeCommand("serve").run([conf_file] + args)

```

Modul rnaseqlyze.worker.stages

```

"""
RNA-Seqlyze Worker Stages

```

```

    -- **this** is where things are actually getting done! :-)
5  """
    import logging
    log = logging.getLogger(__name__)

    import os
10  from os.path import join, exists, isdir, relpath
    from subprocess import Popen, PIPE
    from StringIO import StringIO
    from threading import Thread
    from urllib import quote

15  import pysam

    from Bio import SeqIO
    from Bio.SeqFeature import \
20  SeqFeature, FeatureLocation, ExactPosition

    from psutil import cpu_percent

    from rnaseqlyze import efetch
25  from rnaseqlyze import galaxy
    from rnaseqlyze import ucscbrowser
    from rnaseqlyze import transterm, gb2ptt
    from rnaseqlyze.ucscbrowser import BAMTrack, BigWigTrack, BigBedTrack
    from rnaseqlyze.core.entities import GalaxyDataset
30

    class Operon(object):
        def __init__(self, **kwargs):
            self.__dict__.update(kwargs)
35

        _stages = []
        _stage_conds = {}
        def stage(method):
            """
40  Just a small helper to collect the stages in the order defined.

To add a new stage, simply add a method to :class:`~WorkerStages`.
It will be automatically executed for all new analyses.
            """
45  if method.func_name in _stage_conds:
                method.should_run = _stage_conds[method.func_name]
                del _stage_conds[method.func_name]
            else:
                method.should_run = lambda self: True
50  _stages.append(method)
            return method

        def stage_cond(method):
            """
55  Stage Condition

            - must be declared before @stage

```

```

        - must return true for the @stage of the same name to run
        """
60     _stage_conds[method.func_name] = method

class WorkerStages(object):
    """
    Available attributes:
65
        - self.analysis
        - self.session = DBSession(bind=create_engine(rnaseqlyze.db_url))

    .. note::
70        After changing one of the attributes of the self.analysis object,
        **always** **immediately** call self.session.commit(). Otherwise
        the database will stay locked and the web frontend can't update the ui.

    """
75
    #####
    # Utility Methods & Properties

80    def log_cmd(self, *cmd):
        # can't wait() on subprocess with a timeout, alas start up
        # a 2nd thread to do it and join() on that one with a timeout
        log.info("forking subprocess: $ %s" % ' '.join(map(repr, cmd)))
        proc = Popen(cmd, stdout=self.logfile, stderr=self.logfile)
85        waiter = Thread(target=proc.wait)
        cpu_percent(0, True)
        waiter.start()
        waiter.join(15)
        while waiter.is_alive():
90            log.info("subprocess still running - system load: " +
                    " / ".join("%d%%" % p for p in cpu_percent(0, True)))
            waiter.join(15)
        if proc.returncode != 0:
            raise Exception("%s failed" % (cmd,))
95

    @property
    def srr_name(self):
        return self.analysis.inputfile_base_name

100    @property
    def bam_name(self):
        return "%s %s Mapping" % (self.genbank_record.id, self.srr_name)

    @property
105    def coverage_name(self):
        return "%s %s Coverage" % (self.genbank_record.id, self.srr_name)

    @property
    def hp_terms_name(self):
110        return "%s Hairpin Terminators" % (self.genbank_record.id,)

```

```

@property
def pr_operons_name(self):
    return "%s Predicted Operons" % (self.genbank_record.id,)
115

#
#####
# Stages

120
@stage_cond
def determine_inputfile_type(self):
    return self.analysis.inputfile_uploaded

@stage
def determine_inputfile_type(self):
125
    _8bytes = open(self.analysis.inputfile_path).read(8)
    log.info("first 8 bytes of input data: %r" % _8bytes)
    self.analysis.inputfile_type = (
        'fastq' if _8bytes[0] == '@'
        else 'sra' if _8bytes == 'NCBI.sra' else None)
130
    self.session.commit()
    if not self.analysis.inputfile_type:
        raise Exception("Unknown input data type")

@stage_cond
135
def fetch_srr(self):
    # don't download if private
    # file uploaded or srr already in cache
    return not self.analysis.inputfile_uploaded \
        and not os.path.exists(self.analysis.rnaseq_run.sra_path)

140
@stage
def fetch_srr(self):
    self.analysis.rnaseq_run.download()

@stage_cond
145
def convert_input_file(self):
    return not exists(self.analysis.inputfile_fq_path)

@stage
def convert_input_file(self):
    os.chdir(self.analysis.input_data_dir)
    self.log_cmd("fastq-dump", self.analysis.inputfile_name)
150
    log.debug("created %s" % self.analysis.inputfile_fq_path)

@stage_cond
def fetch_genbank_file(self):
155
    return not exists(self.analysis.genbankfile_path)

@stage
def fetch_genbank_file(self):
    if not exists(self.analysis.genbank_data_dir):
        os.makedirs(self.analysis.genbank_data_dir)
160
    log.info("Fetching '%s' from entrez..." %
        self.analysis.org_accession)
    gb_id = efetch.get_nc_id(self.analysis.org_accession)
    efetch.fetch_nc_gb(gb_id, open(self.analysis.genbankfile_path, "w"))
    log.info("...done")
165

```

```

@stage
def read_genbank_file(self):
    self.genbank_record = SeqIO.parse(open(self.analysis \
                                           .genbankfile_path), "genbank").next()
170     ngenes = sum(1 for f in self.genbank_record.features
                   if f.type == 'gene')
    log.info("genbank file lists %d genes" % ngenes)

@stage_cond
175 def genbank_to_fasta(self):
    return not exists(self.analysis.genbankfile_fa_path)

@stage
def genbank_to_fasta(self):
    log.info("Converting '%s' to fasta format" %
180              self.analysis.genbankfile_name)

    record = self.genbank_record
    saved_id = record.id
    record.id = "chr" # make ucsc browser custom tracks work
    SeqIO.write(record, open(
185         self.analysis.genbankfile_fa_path, "w"), "fasta")
    record.id = saved_id

@stage_cond
def bowtie_build(self):
190     return not exists(join(self.analysis.genbank_data_dir,
                           self.analysis.genbankfile_base_name + ".1.bt2"))

@stage
def bowtie_build(self):
    os.chdir(self.analysis.genbank_data_dir)
195     self.log_cmd("bowtie2-build", self.analysis.genbankfile_fa_name,
                  self.analysis.genbankfile_base_name)

@stage_cond
def tophat(self):
200     return not exists(join(self.analysis.data_dir,
                           "tophat-output", "accepted_hits.bam"))

@stage
def tophat(self):
    os.chdir(self.analysis.data_dir)
205     n_cpus = os.sysconf("SC_NPROCESSORS_ONLN")
    fq = relpath(self.analysis.inputfile_fq_path)
    gb = relpath(join(self.analysis.genbank_data_dir,
                      self.analysis.genbankfile_base_name))
    self.log_cmd("tophat",
210                  "-p", str(n_cpus),
                  "-o", "tophat-output",
                  "--segment-length", "999999999",
                  "--no-coverage-search", "--no-novel-juncs", gb, fq)

215 @stage_cond
def create_coverage_track(self):
    return not exists(join(self.analysis.data_dir, "coverage.bigwig"))

@stage
def create_coverage_track(self):

```

```

220         os.chdir(self.analysis.data_dir)
           # the script automatically converts it's
           # output to bigwig if it finds kent's wigToBigWig
           self.log_cmd("bam_to_wiggle.py", "-o", "coverage.bigwig",
                        "tophat-output/accepted_hits.bam")

225     @stage_cond
       def genbank_to_ptt(self):
           return not exists(join(self.analysis.genbank_data_dir,
                                   self.genbank_record.id + ".ptt"))

230     @stage
       def genbank_to_ptt(self):
           ptt_name = self.genbank_record.id + ".ptt"
           ptt_path = join(self.analysis.genbank_data_dir, ptt_name)
           os.symlink(ptt_name, join(self.analysis.genbank_data_dir, "chr.ptt"))
235           log.debug("converting %s to ptt" % self.analysis.genbankfile_name)
           ptt_file = open(ptt_path, "w")
           gb_file = open(self.analysis.genbankfile_path)
           gb2ptt.gb2ptt(gb_file, ptt_file)
           ptt_file.close()
240           gb_file.close()

           @stage_cond
       def transterm_hp(self):
           return not exists(join(self.analysis.data_dir,
                                   "hp_terminators.bigbed"))

245     @stage
       def transterm_hp(self):
           os.chdir(self.analysis.data_dir)
           log.debug("running transterm")

250           tt_out = open("transterm_hp.out", "w+")
           # --min-conf=n n is the cut-off confidence value,
           #               between 0 and 100, the default is 76
           tt_args = ("--min-conf=47",
255                       self.analysis.genbankfile_fa_path,
                       relpath(join(self.analysis.genbank_data_dir, "chr.ptt")))
           transterm.run(tt_args, out=tt_out, err=self.logfile)
           tt_out.seek(0)
           # keep a copy in memory
260           self.hp_terminators = list(transterm.iterator(tt_out))
           tt_out.seek(0)
           log.info("found {0} possible hairpin terminators"
                    .format(len(self.hp_terminators)))

           # create a bed track
265           bed_file = open("hp_terminators.bed", "w")
           transterm.tt2bed(tt_out, bed_file)
           bed_file.close()
           tt_out.close()

270           log.debug("running bedToBigBed")

           # convert it to bigBed
           chrs = open("chrom.sizes", "w")

```

```

275     chrs.write("chr %d" % len(self.genbank_record.seq))
    chrs.close()
    self.log_cmd("bedToBigBed", "hp_terminators.bed",
                 "chrom.sizes", "hp_terminators.bigbed")

    @stage
280     def predict_operons(self):

        # extract the coverage data from the bam track created by tophat
        bam_path = join(join(self.analysis.data_dir,
                             "tophat-output", "accepted_hits.bam"))
285     if not exists(bam_path + ".bai"):
        pysam.index(bam_file)

        self.max = 0
        self.covered = 0
290     self.coverage = [0] * len(self.genbank_record.seq)

        sam_reader = pysam.Samfile(bam_path, "rb")
        chrom, length = sam_reader.references[0], sam_reader.lengths[0]

295     assert chrom == "chr" and length == len(self.genbank_record.seq), (
        "Something went badly wrong"
        " -- the bam track or genbank file could be corrupted...")

        for base in sam_reader.pileup(chrom, 0, length):
300             self.covered += 1
            if base.n > self.max:
                self.max = base.n
            self.coverage[base.pos] = base.n

305     if not self.covered:
        raise Exception("Not a valid bam file")

        log.debug("maximum coverage: %d" % self.max)
        log.debug("number of bases covered by short reads: %d/%d" % (
310             self.covered, len(self.genbank_record.seq)))

        # available objects at this point
        # -----
        #
315     # - self.genbank_record: Biopython SeqIO.parse()d genbank file
        #
        # - self.coverage: [n,n,n,n,...] / len = len(self.genbank_record.seq)
        #
        # - self.max: max(n)
320     #
        # - self.hpterminalators: ((id, begin, end, strand, confidence), ...)
        #                          str, str, str, str (1/-), int
        #
325     # FIXME: do some magic here
    self.operons = Operon(begin=0, end=100, strand=1, confidence=10),
    self.operons = Operon(begin=200, end=300, strand=1, confidence=50),

```



```

self.operons = Operon(begin=400, end=500, strand=1, confidence=100),

330     # create a bed track
    track_name = "rna-seqlyze-operon-predictions"
    os.chdir(self.analysis.data_dir)
    bed_file = open(track_name + ".bed", "w")
    for i, o in enumerate(self.operons):
335         begin, end = str(o.begin), str(o.end)
        rgb_color = ','.join((str(100 - int(o.confidence)),)*3)
        print >> bed_file, '\t'.join((
            'chr', begin, end,
            'OPERON_%d' % i, str(o.confidence),
340         '+' if o.strand > 0 else '-', begin, end, rgb_color
        ))
    bed_file.close()

    # convert it to bigBed
345     # chrom_sizes already generated during "transterm_hp"
    self.log_cmd("bedToBigBed", track_name + ".bed",
                 "chrom.sizes", track_name + ".bigbed")

@stage
350 def upload_track_data(self):

    # FIXME: names are not unique on galaxy:
    # is "%s_%s" % (srr_name, self.analysis.org_accession) good enough ?

355     if not self.analysis.galaxy_bam:
        bam_path = join(self.analysis.data_dir,
                        "tophat-output", "accepted_hits.bam")
        log.info("uploading accepted_hits.bam to galaxy")
        self.analysis.galaxy_bam = GalaxyDataset(
360             id=galaxy.upload(open(bam_path), self.bam_name))
        log.info("...done - id: %s" % self.analysis.galaxy_bam.id)
        self.session.commit()

    if not self.analysis.galaxy_coverage:
365         coverage_path = join(self.analysis.data_dir, "coverage.bigwig")
        log.info("uploading coverage.bigwig to galaxy")
        self.analysis.galaxy_coverage = GalaxyDataset(
            id=galaxy.upload(open(coverage_path), self.coverage_name))
        log.info("...done - id: %s" % self.analysis.galaxy_coverage.id)
370         self.session.commit()

    if not self.analysis.galaxy_hp_terms:
        hp_terms_path = join(self.analysis.data_dir,
                            "hp_terminators.bigbed")
375         log.info("uploading hp_terminators.bigbed to galaxy")
        self.analysis.galaxy_hp_terms = GalaxyDataset(
            id=galaxy.upload(open(hp_terms_path), self.hp_terms_name))
        log.info("...done - id: %s" % self.analysis.galaxy_hp_terms.id)
        self.session.commit()

380     if not self.analysis.galaxy_pr_operons:

```

```

        track_filename = "rna-seqlyze-operon_predictions.bigbed"
        pr_operons_path = join(self.analysis.data_dir, track_filename)
        log.info("uploading %s to galaxy" % track_filename)
385         self.analysis.galaxy_pr_operons = GalaxyDataset(
            id=galaxy.upload(open(pr_operons_path), self.pr_operons_name))
        log.info("...done - id: %s" %
                self.analysis.galaxy_pr_operons.id)
        self.session.commit()
390
    @stage
    def create_and_upload_hg_text(self):
        """
        ``hgt.customText`` is a parameter of the UCSC
395        "hgTracks" genome browser application that makes it
        possible to share "custom tracks" via a url.

        The value of the ``hgt.customText`` parameter is itself
        an URL. The shareable "custom tracks url" is therefore an
400        URL that contains another URL. The other url must be "escaped"
        for this to work. That actually happens in
        :meth:`~rnaseqlyze.core.analysis.AnalysisMixins.hg_url`.

        The details are explained here:
405        http://genome.ucsc.edu/goldenPath/help/customTrack.html#SHARE
        """

        if self.analysis.galaxy_hg_text:
            return
410

        tracks = []

        # FIXME: this cries for refactoring -- with logging!

415        # bam track (mapping)
        bam_url = "https://" + galaxy.hostname \
            + galaxy.ucsc_bam_path_template \
                .format(dataset=self.analysis.galaxy_bam.id)
        tracks.append(BAMTrack(url=bam_url,
420                             name="RNA-Seqlyze | %s" % self.bam_name))

        # bigwig track (coverage)
        coverage_url = "https://" + galaxy.hostname \
            + galaxy.dataset_display_url_template \
                .format(dataset=self.analysis.galaxy_coverage.id)
425        tracks.append(BigWigTrack(url=coverage_url,
                                    name="RNA-Seqlyze | %s" % self.coverage_name))

        # bigbed track (terminators)
430        hp_terms_url = "https://" + galaxy.hostname \
            + galaxy.dataset_display_url_template \
                .format(dataset=self.analysis.galaxy_hp_terms.id)
        tracks.append(BigBedTrack(url=hp_terms_url,
435                                 name="RNA-Seqlyze | %s" % self.hp_terms_name))

```

```

# bigbed track (predicted operons)
pr_operons_url = "https://" + galaxy.hostname \
    + galaxy.dataset_display_url_template \
    .format(dataset=self.analysis.galaxy_pr_operons.id)
440 tracks.append(BigBedTrack(url=pr_operons_url,
    name="RNA-Seqlyze | %s" %
        self.pr_operons_name))

track_file = StringIO()
445 track_file.write('\n'.join(tracks))
track_file.seek(0)
self.analysis.galaxy_hg_text = GalaxyDataset(
    id=galaxy.upload(track_file,
        "UCSC Tracks Analysis%d.txt" %
450 self.analysis.id))

self.session.commit()

@stage
def create_genbank_file(self):
455 """
    Create a genbank file containing

    For more documentation on how to create new features, visit

460 - http://biopython.org/\
    DIST/docs/api/Bio.SeqRecord.SeqRecord-class.html#__getitem__
    - http://biopython.org/\
    DIST/docs/api/Bio.SeqFeature.SeqFeature-class.html

465 - http://www.ebi.ac.uk/\
    embl/Documentation/FT_definitions/feature_table.html
    """

log.info("augmenting genbank file %s with putative operons" %
470 self.analysis.genbankfile_name)

for i, o in enumerate(self.operons):
    location = FeatureLocation(ExactPosition(o.begin),
        ExactPosition(o.end))
475 self.genbank_record.features.append(
    SeqFeature(location,
        type='mRNA',
        strand=o.strand,
        qualifiers=dict(
480     note='putative, confidence %d%%' % o.confidence,
        operon='rnas-%d' % i)))

self.genbank_record.features.sort(
    key=lambda f: f.location.start.position)
485 xgb_file = open(self.analysis.xgenbankfile_path, "w")

SeqIO.write(self.genbank_record, xgb_file, "genbank")
#

```

```

490 #####

assert not _stage_conds, "@stage_cond's must be declared before @stage's"
WorkerStages.stages = _stages
del _stages, _stage_conds

```

3.2.6 Javascript Code

File *rnaseqlyze.js*

```

/*
 * RNA-seqlyze javascript routines
 */

5 $(function() {

    // global variables and helpers
    // -----

10    // ___ el ___
    // from http://joestelmach.github.com/laconic/
    window.el = $.el;

    // log.info() and log.debug()
15    window.log = {
        'info': function() {
            console.log.apply(console, arguments);
        },
        'debug': function() {}
20    }
    if (rnaseqlyze_debug)
        window.log.debug = window.log.info;

    // page initializaion
    // -----

25    // use bootstrap's
    // "scrollspy" plugin
    // -- patched version - see https://github.com/twitter/bootstrap/pull/3829
30    $(window).scrollspy({
        // offset: 200,
        wrap: $('#wrap')[0],
    });

35    // Generally useful stuff
    // -----

    // based on http://stackoverflow.com/a/4673436
40    String.prototype.format = function() {
        var i = 0; args = arguments;
        return this.replace(/{} /g, function() {

```

```

        return args[i++];
45    });
    };

});

50 // http://stackoverflow.com/a/7531350
jQuery.fn.extend({
    scrollToBottom: function () {
        var top = $(this).offset().top;
        var offtop = top - 250 + $(this).height();
55    jQuery('html,body').animate({scrollTop: offtop}, 100);
    },
});

```

File *rnaseqlyze-create.js*

```

/*
 * RNA-seqlyze javascript routines
 *
 * for the "create" page
5 */

$(document).ready(function() {

    /*
10    * Toggle input type
    */

    $('#input_type_radio').click(function(event) {
        if ($(event.target).hasClass("srr")) {
15        // user chose the "Data File" option
            $('#sra-controls').hide();
            $('#srr-controls').show();
        } else if ($(event.target).hasClass("sra")) {
            // user chose the "SRR Identifier" option
20        $('#srr-controls').hide();
            $('#sra-controls').show();
        } // else
            // what the...
    });
25 $('#input_type_radio .srr').click();

    /*
    * discretionary #pairendlenControls
    */
30

    function maybe_show_pairendlen_controls() {
        if ($('#pairendedInput').attr('checked'))
            $('#pairendlenControls').show();
        else
35        $('#pairendlenControls').hide();
    }
}

```

```

maybe_show_pairedlen_controls();
$('#pairedlenInput').change(maybe_show_pairedlen_controls);

40

/*
 * Toggle organism input type
 */

45

$('#org_type_radio').click(function(event) {
    if ($('#event.target').hasClass("title")) {
        // user chose "Title"
        $('#genbankfile-controls').hide();
        $('#org_title-controls').show();
50    } else if ($('#event.target').hasClass("file")) {
        // user chose "Genbank File"
        $('#org_title-controls').hide();
        $('#genbankfile-controls').show();
55    } // else
        // what the ...
    });
    $('#org_type_radio .title').click();

60

/*
 * Organism input autocompletion
 */

var organisms = new Array();

65

$.ajax({
    url: "rest/organisms",
    dataType: "json",
    success: function(data) {
70        // No idea yet what to do with those that have multiple
        // accessions listed in 'genome' -- filter them out here for now
        //
        // see ftp://ftp.ncbi.nih.gov/genomes/GENOME_REPORTS/prokaryotes.txt
        //
75        _(data).each(function(org) {
            if (org.acc.indexOf(",") >= 0)
                return;
            organisms.push("{} ({} / {})".format(
                org.title, org.db, org.acc)); }); },
80    });

$('#organismInput').typeahead({
    source: organisms,
    });

85

/*
 * plupload -- from src/plupload/examples/custom.html
 */

90

var uploads = {
    'inputfile': {},

```

```

    'genbankfile': {}
};

95  var context = function(name) {

    var self = this;

    var options = {
100      url:      'upload',
      runtimes:  'html5,gears,flash,silverlight,browserplus,html4',
      browse_button:  name + '_browse',
      drop_element:   name + '_progress',
      multipart_params: { 'type': name,
105        'session': upload_session },
      flash_swf_url:   path_js + '/plupload.flash.swf',
      silverlight_xap_url: path_js + '/plupload.silverlight.xap',
    };

110  var events = {
    // 'Init': function(up, params) {
    //      $('#' + name + '_progress .filestatus').text(
    //          "Current runtime: " + params.runtime);
    // },
115  'FilesAdded': function(up, up_files) {
    // remove all other files already present
    // plupload features multiple files in one widget
    // we have two widgets and one name per widget
    up.splice();
120    self.active = true;
    log.debug("FilesAdded", up.files, up_files);
    $('#' + name + '_progress .filestatus').text(
        up_files[0].name +
        ' (' + plupload.formatSize(up_files[0].size) + ')');
125  },
    'UploadComplete': function(up, up_files) {
        self.complete = true;
        for (nam in uploads)
            if (uploads[nam].active)
130                if(!uploads[nam].complete)
                    return;
        log.debug("UploadComplete", "go");
        $('#create_form').submit();
    },
135  'UploadProgress': function(up, up_file) {
        $('#' + name + '_progress .bar').css(
            "width", up_file.percent + '%');
        // $('#' + name + '_progress .filestatus').text(
        //      up_file.percent + '%');
140  },
    };

    this.active = false;
    this.complete = false;
145

```

```

    var up = this.up = new plupload.Uploader(options);

    $('#' + name + '_progress').click(function() {
    $('#' + name + '_browse').click();
150 });

    $('#create_form_submit').click(function() {
        for (nam in uploads)
            if (uploads[nam].active)
155             { up.start(); return false; }
    });

    for (x in events)
        up.bind(x, events[x]);
160

    log.debug("debug");

    up.init();
};
165

for (name in uploads)
    uploads[name] = new context(name);
});

170 // vim: et:sw=4

```

File rnaseqlyze-analysis.js

```

/*
 * RNA-seqlyze 'analysis' view javascript
 */

5 // backbone.js Models
// -----
//
// -> http://backbonejs.org/#Model

10 // The Analysis
window.Analysis = Backbone.Model.extend({
    urlRoot: "../rest/analyses",
    initialize: function () {
        this.files = new DataDirListing();
15         this.files.analysis = this;

        this.stage_logs = new StageLogList();
        this.stage_logs.analysis = this;

20         // "cascade": update files list
        this.bind("change:data_dir_state", function (self) {
            self.files.fetch({add: true});
        });
        // "cascade": update stage_logs
25         this.bind("change:stage_logs_state", function (self) {

```



```

        var len = self.stage_logs.size();
        self.stage_logs.fetch({
            add: true,
            // The last stage_log is the current one and updates frequently.
            // It's id stays the same though and which causes backbone.js
            // to regard it as a duplicate and drop it. But a copy of the
            // ajax response is passed to the success callback. So we pick
            // the changed log text from there and fire a "change" event
            // by set()ting the 'text' attribute of the affected model.
            success: function (stage_logs, rsp) {
                if (len)
                    stage_logs.at(len-1).set('text', rsp[len-1].text);
            },
        });
    });
}
});
// log output of one stage
window.StageLog = Backbone.Model.extend({
    defaults: {
        stage: null,
        text: null,
    },
    idAttribute: "id",
});
// log output of all stages
window.StageLogList = Backbone.Collection.extend({
    model: StageLog,
    url: function () {
        return this.analysis.url() + "/logs";
    },
});
// a model for the files
window.DataDirFile = Backbone.Model.extend({
    defaults: {
        path: null,
    },
    idAttribute: "path",
});
// and for a collection of files
window.DataDirListing = Backbone.Collection.extend({
    model: DataDirFile,
    url: function () {
        return this.analysis.url() + "/files";
    },
});

// note1:
//
// Concerning the above code:
// It might have been simpler to work the files list right into the
// analysis model on the server and stick to one model here.
// But then again, there is no harm in doing it like this, because
// now the files list is more independent and could for

```

```

80 // example also be displayed on a page of its own.

/* note2:
 *
85 * In the code below,
 *
 * "el" is defined in rnaseqlyze.js as "$.el", which is defined
 * in laconic.js - see http://joestelmach.github.com/laconic/
 *
90 * "this.$el"
 * "this.el" are the view's (jQuery wrapped) DOM element in the
 * backbone.js architecture - see http://backbonejs.org/#View-el
 *
 * "render().el" is also the view's "el" and works because we always
95 * "return this;" from render() - see http://backbonejs.org/#View-render
 */

// Two Views showing different details about the analysis
// -----

100 // These render the "Processing" and "Results" section on the
// analysis page. The Processing view is displayed above the Results view.
//
// -> http://backbonejs.org/#View

105 // The "Processing" section
window.ProcessingView = Backbone.View.extend({

  initialize: function () {
110    this.model.bind("change", this.change, this);
    this.stage_logs = (
      new StageLogListView({model: this.model.stage_logs}).render().el);
  },
  change: function (model, value, options) {
115
    // just re-render the whole thing for now
    this.$el.empty();
    this.render();

120
    // remove the busy indicator when finished
    if (model.get('finished'))
      $('#spinner').remove();

    // make scrollspy refresh it's coordinates
    // because the page size has likely changed
125    $(window).scrollspy('refresh');
  },
  render: function () {
    // toJSON doesn't really do much besides turning
    // the model.attributes into a useable object
    // see http://backbonejs.org/#Model-toJSON
130    var analysis = this.model.toJSON();

```

```

    this.$el.append(
135       el.h2("Processing")
    );

    this.$el.append(el.div(
140       el.h3("Input Check")
    ,
        analysis.inputfile_uploaded ?
            el.p("Type of input: ",
                analysis.inputfile_type ?
                    el.strong(analysis.inputfile_type) :
145                     el.span("not detected"))
            :
            null
    ,
        analysis.inputfile_header ?
150         el.p("First read in input data: ",
            el.pre(analysis.inputfile_header))
            :
            null
    ));

155
    this.$el.append(el.div(
        el.h3("Stage Logs")
    ,
160         this.stage_logs
    ));

    if (analysis.error)
        this.$el.append(
165         el.div({class: "alert alert-error"},
            el.h4({class: "alert-heading"},
                "An error occured while analyzing the data"),
            analysis.error));

170     return this;
    },
    });

    // The monospaced stage log blocks
175 window.StageLogListView = Backbone.View.extend({
    initialize: function () {
        this.model.bind("add", this.add, this);
    },
    add: function (model) {
180         this.$el.append(
            new StageLogView({model: model}).render().el
        );
        if (!this.model.analysis.get('finished')) {
            this.$el.contents().find("pre")
185                 .not(':last').css('background-color', '')
                .prevObject.last().css('background-color', '#ddf');
            this.$el.contents().last().scrollTopBottom();
        }
    }
});

```

```

    }
    $(window).scrollspy('refresh');
190   },
  });

  // _One_ monospaced stage log block
  window.StageLogView = Backbone.View.extend({
195    initialize: function () {
      this.model.bind("change", this.change, this);
    },
    change: function (model, options) {
      this.$el.children("pre").text(model.get('text'));
200      this.$el.scrollToBottom();
      $(window).scrollspy('refresh');
    },
    render: function () {
      var log = this.model.toJSON();
205      this.$el.attr('id', log.stage);
      this.$el.append(el.h4(log.stage));
      this.$el.append(el.pre(log.text));
      return this;
    },
210  });

  // The "Results" section
  window.ResultsView = Backbone.View.extend({
    initialize: function () {
215      this.model.bind("change", this.change, this);
      this.model.files.bind("add", this.fileadd, this);
    },
    change: function (model, value, options) {
      this.$el.empty();
220      this.render();
      $(window).scrollspy('refresh');
    },
    fileadd: function () {
      var augmented_gb = this.model.files.find(function (file) {
225        return file.get('path').match(/augmented\.gb$/);
      });
      if (!this.augmented_gb) {
        this.augmented_gb = augmented_gb;
        this.$el.empty();
230        this.render();
      }
    },
    render: function () {
      var analysis = this.model.toJSON();
235      if (analysis.hg_url || this.augmented_gb) {
        this.$el.append(el.h2("Results"));
        var $ul = $(el.ul())
        this.$el.append($ul[0]);
        if (this.augmented_gb) {
240          var href = '_id + "/files/' + this.augmented_gb.get('path');
          $ul.append(el.li(

```

```

        el.a({href: href},
              "Augmented Genbank File"))));
    }
245     if (analysis.hg_url) {
        $ul.append(el.li(
            el.a({href: analysis.hg_url},
                "Link to custom tracks in UCSC browser"),
            el.p("It might take a minute until the tracks become " +
250                "available.", el.br(),
                "As soon as the last few items ",
                el.a({href: galaxy_history_url}, "here"),
                " turn green it should work.")));
    }
255 }
    return this;
},
});

260 // A View displaying the list of files associated with
// this analysis available on the server (log files, mostly).
// This is currently rendered inside the "Processing" section above.
window.DataDirView = Backbone.View.extend({
    initialize: function () {
265         this.model.bind("reset", this.reset, this);
        this.model.bind("add", this.add, this);
    },
    reset: function (model, value, options) {
        this.$el.empty();
270         this.render();
    },
    render: function () {
        this.$el.append(el.h2("Data Directory"));
        var ul = el.ul();
275         this.$ul = $(ul);
        this.$el.append(ul);
        $(window).scrollspy('refresh');
        return this;
    },
    add: function(model) {
280         this.$ul.append(
            new DataDirFileView({model: model}).render().el
        );
        $(window).scrollspy('refresh');
285     },
});
// An View, that renders one file
window.DataDirFileView = Backbone.View.extend({
    el: "<li>",
290     render: function (model) {
        var file = this.model.toJSON();
        var href = _id + '/files/' + file.path;
        this.$el.html(el.a({href: href}, file.path));
        return this;
295     }
});

```

```

});

// Initialization
// -----
300

$(document).ready(function () {

    // the id of the displayed analysis
305    _id = _(window.location.pathname.split('/')).last();

    // create a backbone.js Model
    // with an associated Collection
    analysis = new Analysis({
310        id: _id,
    });

    // create two backbone.js views for the
    // analysis, render and insert them into the DOM
315    $('#processing').html(
        new ProcessingView({model: analysis}).render().el
    );
    $('#results').html(
        new ResultsView({model: analysis}).render().el
320    );
    $('#datadir').html(
        new DataDirView({model: analysis.files}).render().el
    );

325    /*
    // uncomment this to see what's going on in backbone.js
    if (rnaseqlyze_debug) {
        analysis.bind("all", function (event) {
            log.debug("analysis", arguments);
330        });
        analysis.files.bind("all", function (event) {
            log.debug("analysis.files", arguments);
        });
        analysis.stage_logs.bind("all", function (event) {
335            log.debug("analysis.stage_logs", arguments);
        });
    }
    */

340    // update the models until the analysis is finished
    var update = function () {
        // check at the beginning and not at the end
        // because the fetch() calls are asynchronous
        if (analysis.attributes.finished)
345            return;
        analysis.fetch();
        // log.debug("analysis.fetch()");
        window.setTimeout(update, 7000); // re-update in 7 seconds
    }
}

```

```
350     update();  
    });
```

4 Ausblick

4.1 Komplettierung

4.2 Inbetriebnahme

5 Anhang

5.1 Glossar

RNA-seq Next-generation Sequenzierungs-Technologie angewendet auf das Profiling kompletter Transkriptome

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetur.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.