Educational Laboratory Website Manual

March 9, 2018

Contents

0.1 Introduction

0.2 Making Changes to Website

All changes to the website should be made on the development space on slug (/usr/local/master/pjl-web). The only exception to this rule is that the equipment database equipmentDB.xml can be modified live by using the inventory website in edit mode. Because of this it is important to sync the development and live version of the website before making changes.

0.2.1 Sync Live Version with Development Space

1. Sync web-server to github using the procedure outlined in section 0.2.3. Note that for the web server "watt" the /path/to/web-folder is /var/www/html.

2. Sync development space to github using the procedure outlined in section 0.2.3. Note that for the development version on "slug" the path is /usr/local/master/pjl-web.

3. Sync the web server from the github using

- admin-user@computer /var/www/html
  $ sudo git pull origin master

4. Development space is now ready for editing.

0.2.2 Order of Making Changes

1. Sync live version and development space
2. Changes Inventory
3. Repository
4. Update live version on web server

0.2.3 Procedure for updating PJL github

1. admin-user@computer /path/to/web-folder
   $ sudo git add .
2. admin-user@computer /path/to/web-folder
   $ sudo git commit -m upload latest “equimpnetDB.xml”
3. admin-user@computer /path/to/web-folder
   $ sudo git pull origin master
4. admin-user@computer /path/to/web-folder
   $ sudo git push origin master
   Username for ‘https://github.com’: admin-user
   Password for ‘https://admin-user@github.com’: ********

0.3 Repository

0.3.1 Directory structure

At top most level is a folder called “repository” that contains all experiment related documents.

At the second level all the files are organized by lab experiment. Each experiment has a folder that is labeled with a naming scheme where the first four characters are the unique identifier number, followed by the name of the lab. The lab name should be descriptive of the experiment itself. In this folder is also a folder called “Support_Docs” that contains any documents useful for the experiment, but not actually used to generate the student document.

At the third level files are organized into versions. Each folder follows a naming scheme where the the first four characters are the unique lab identifier number, followed by "PHYS" followed by the Course Number followed by a two character semester identifier, followed by the year. Each folder contains all the file used to generate the pdf given to students in the course, semester, and year as identified in the folder label.

Directory structure sample.

```
/repository
 ├─ 0072-Nuclear-Decay
 │   └─ 0001-PHYS123FA2017
 │       ├─ lab.tex
 │       ├─ photo.jpg
 │       └─ student.pdf
 └─ Support_Docs
```

Documents can only be added to the repository if they meet the following criteria.

1. The files include the pdf given to students to be used in their course work.
2. All files need to generate the pdf are included.

0.3.2 Adding a new lab to the repository

Before beginning ensure that all equipment used in the new experiment are in the lab inventory, and have equipment ID number.

1. Create a folder for the new lab (example "new-lab-folder"), and place all files for generating student pdf, and the student pdf in new-lab-folder
2. Inside new-lab-folder make a directory called "Support_Docs", and put all documents relevant to lab, but not needed for generation of pdf into it. This might include research papers, sample data, Excel spreadsheets, etc.
3. Inside new-lab-folder create a file called info.csv and add all relevant information to the file.

Listing 1: The file info.csv needs to be in this form. See below for more details xleftmargin

```
Name,Vectors-in-One-and-Two-Dimensions
Type,Labatorial
Disciplines,Newtonian Mechanics,Optics
Topics,Collisions,Momentum
Semester,Fall
Year,2017
Course,325
Equipment,0004-(1)-[0002],0050-(4)
Software,ImageJ,Canon
```

```
PDF, title.pdf
```

Any text that is **Bold** must be configured for each lab. text in *italic* are notes that must be removed. Standard text must be left as is.

- Name, **Name as to be Seen on Website** *Use Standard Title Capitalize Convention.*
- Type, **Type** *Must be either Lab or Labatorial.*
- Disciplines, **Discipline1, Discipline2** *Disciplines must comma separated be taken from the approved list* **Need location of this list.**
- Topics, **Topic1, Topic2** *Topics must comma separated be taken from the approved list* **Need location of this list.**
- Semester, **Semester** *Winter, Spring, Summer, or Fall*
- Year, **Year** *: Four digits.*
- Course, **Course Number** *Three digit number corresponding to the course the experiment was used in.*
- Equipment, **equipID-(Amount)-[alternate equipID], equipID-(Amount)** *equipID is four digit code of equipment in inventory, Amount is how many are needed, alternate ID is the four digit code of equipment in inventory that can be used if the primary unit is not available. IDs amounts and alternate IDs separated by "-", and items in equipment list separated by ","*
- Software, **Software1, Software2** *Name of all software needed. Must be software from the list of supported software* **Need location of this list**
- PDF, **PDF exact Name** *This needs to be the exact name of the student pdf*

4. Run "addNewLab.py /path/to/new-lab-folder" from development space.

0.3.3 Repository xml template

```
<Labs>
    <Lab labId="0001">
        <Name />
        <Disciplines>
            <Discipline />
            ...
        </Disciplines>
        <Topics>
            <Topic />
            ...
        </Topics>
        <Versions>
            <Version>
                <Path />
```

```
                    <Semester />
                    <Year />
                    <Course />
                    <Directory />
                </Version>
                ...
            </Versions>
            <Equipment>
                <Item id="0001">
                    <Name />
                    <Amount />
                </Item>
                ...
            <Equipment />
            <Type />
            <SupportDocs>
                <Doc>
                    <Name />
                    <Path />
                </Doc>
                ...
            </SupportDocs>
            <Software>
                <Name />
                ...
            </Software>
        </Lab>
        ...
    </Labs>
```

0.4 Inventory

0.4.1 Inventory structure

Each item in the inventory should have a unique identifier number, and a unique name. Each item has a place for any number of manuals, and one picture.

0.4.2 Adding New Equipment

1. Compile all rel

Listing 2: The file info.csv needs to be in this form. language

```
name; Unique Name of Kit, is_kit; False, kit ; item1 ; item2 ; item3 (2)
name; Unique Name of Equipment
```

0.4.3 Adding photos of equipment

0.5 Scripts

0.5.1 Add New Lab - addNewLab.py

```
#!/usr/bin/python3

'''Adds version from a csv file and adds them to the repository'''

from pjlDB import *
import csv, argparse, os
import xml.etree.ElementTree as ET

'''Folders and file used.'''
root = "/usr/local/master"
eqdb = EquipDB(root + "/pjl-web/data/equipmentDB.xml")
labdb = LabDB(root + "/pjl-web/data/labDB.xml")
infoFile = "info.csv"
semesterDict = {"Winter": "WI", "Spring": "SP", "Summer": "SU", "Fall": "FA" }


'''Collects information on the lab to add, and formats it to be used by pjlDB.py'''
class NewLabInfo():


        def __init__(self,labSource,newLabObj):
                self.id_num = newLabObj.id_num
                self.labFolder = self.id_num + "-" + labSource.split("/")[-2]
                self.rawInfo = self._collectInfo(labSource,infoFile)
                self.name = "␣".join(self.rawInfo["name"][0].split("-"))
                self.name = "␣".join(self.name.split())
                self.lab_type = self.rawInfo["type"][0].capitalize()
                self.equipment = self.equipInfo()
                self.software = self.rawInfo.get("software","")
                self.semester = self.rawInfo["semester"][0].capitalize()
                self.semester = "␣".join(self.semester.split())
                self.year = self.rawInfo["year"][0]
                self.year = "␣".join(self.year.split())
                self.course = self.rawInfo["course"][0]
                self.versionDr = self.id_num + "-PHYS" + self.course + semesterDict[self.semester] + self.
                self.versions = self.makeVersionDict()
                self.topics = self.optionalTag("topics")
                self.disciplines = self.optionalTag("disciplines")


        def optionalTag(self,typ):
                checkForNone = self.rawInfo.get(typ)
                if checkForNone != None:
                        if not len(checkForNone) == 0:
                                if not checkForNone[0] == '':
                                        checkForNone = [i.strip('␣') for i in checkForNone]
                                        for i in checkForNone:
                                                if i == '':
                                                        checkForNone.remove(i)
                                        return checkForNone
                                else:
                                        return []
                        else:
                                return []
                else:
                        return []
```

```
    '''Checks to see if info.csv file exists'''
    def _collectInfo(self,dr,filename):
            self.infoPath = dr + filename
            if os.path.isfile(self.infoPath):
                    csvContents = self._convertCSVtoDic()
            else:
                    print(i + "_is_missing_the_configuration_file_info.csv")
                    print("exiting...")
                    exit()
            return csvContents


    '''Turns data in info.csv into dictionary'''
    def _convertCSVtoDic(self):
            labInfoDict = {}
            with open(self.infoPath, "r") as o:
                    labInfo = csv.reader(o)
                    for i in labInfo:
                            category = i[0].lower()
                            contents = i[1:]
                            labInfoDict[category] = contents
            return labInfoDict


    '''Makes a dictionary of version to feed into pjlDB.py'''
    def makeVersionDict(self):
            versions = []
            version = {}
            version["path"] = "/data/repository/" + self.labFolder + "/" + self.versionDr \
            + "/" + self.rawInfo["pdf"][0]
            version["semester"] = self.semester
            version["course"] = "PHYS_" + self.course
            version["year"] = self.year
            directory = "/data/repository/" + self.labFolder + "/" + self.versionDr
            version["directory"] = directory
            versions.append(version)
            return versions


    '''Finds the name of equipment from inventory, and returns list'''
    def equipInfo(self):
            equipItems = []
            equipRawInfo = self.rawInfo["equipment"]
            for i in equipRawInfo:
                    itemDict = {}
                    eqid = i.split("-")[0].strip()
                    if eqdb._idExistsAlready(eqid) == True:
                            itemDict["id"] = eqid
                            itemDict["name"] = eqdb.getItem(eqid).name
                            itemDict["amount"] = i.split("(")[1].split(")")[0]
                            if "[" in i:
                                    altid = (i.split("[")[1].split("]")[0])
                                    itemDict["alt-name"] = eqdb.getItem(altid).name
                                    itemDict["alt-id"] = altid
                            else:
                                    itemDict["alt-name"] = ""
```

```
                                        itemDict["alt-id"] = ""
                        equipItems.append(itemDict)
                return equipItems


def emptyList(tagInfo):
        for i in tagInfo:
                if i == "":
                        tagInfo.remove(i)
        if len(tagInfo) == 0:
                return True
        else:
                return False


'''Adds information to new xml entry'''
def addLabToXML(labInfo,newlab,labdb):
        newlab.name = labInfo.name
        newlab.lab_type = labInfo.lab_type
        #if not emptyList(labInfo.disciplines):
        newlab.disciplines = labInfo.disciplines
        #if not emptyList(labInfo.topics):
        newlab.topics = labInfo.topics
        newlab.equipment = labInfo.equipment
        newlab.software = labInfo.software
        newlab.versions = labInfo.versions
        if testing == True:
                print(newlab.name)
                print(newlab.lab_type)
                print(newlab.disciplines)
                print(newlab.topics)
                print(newlab.equipment)
                print(newlab.software)
                print(newlab.versions)

        labdb.addLab(newlab)
'''Collects info on the source and destination directories of new lab'''
def getDrMoveInfo(labInfo,repository):
        drInfo = {}
        drInfo["originDr"] = i
        drInfo["labDr"] = root + "/labs/repository/" + labInfo.labFolder
        drInfo["versionDr"] = drInfo["labDr"] +"/" + labInfo.versionDr#[0]["directory"]
        return drInfo

def makeDr(newDr):
        print("newDr is " + newDr)
        if not os.path.isdir(newDr):
                os.system("mkdir " + newDr)
        else:
                print("Lab folder " + newDr + " Already Exists.")
                print("If this is a new version of an existing lab try running addVersions.py")
                print("Exiting...")
                exit()

def checkForDr(existingDr):
        if not os.path.isdir(existingDr):
                print("Source folder " + existingDr + " Does Not Exists. Exiting ...")
                exit()
        else:
                return True

def moveDr(origin,destination):
        if checkForDr(origin) and checkForDr("/".join(destination.split("/")[:-1])):
                os.system("mv␣" + origin + "␣" + destination)


'''Reads in location of folder for lab to add'''
parser = argparse.ArgumentParser()
parser.add_argument('source', nargs='+', help='enter␣path␣of␣lab␣folder(s)␣to␣add.␣\
␣␣␣␣␣␣␣␣Ex␣addNewLab␣/path/to/folder1␣~/path/to/folder2')
parser.add_argument('-t', '--test', help='test␣adding␣to␣xml␣without␣moving␣folders', action='store_true')
args = parser.parse_args()
labDr = args.source
testing = args.test

'''Adds new lab to labDB.xml'''
newDrInfo = []
for i in labDr:
        newLab = labdb.newLab(labdb.new_id)
        labInfo = NewLabInfo(i,newLab)
        addLabToXML(labInfo,newLab,labdb)
        newDrInfo.append(getDrMoveInfo(labInfo,root))
labdb.save(root + "/pjl-web/dev/labDB.xml", ignore_validation=False)
valid = labdb.validateFull()

if testing != True:
        if valid == True:
                for i in range(0,len(labDr)):
                        makeDr(newDrInfo[i]["labDr"])
                        moveDr(newDrInfo[i]["originDr"],newDrInfo[i]["versionDr"])
                        origin = newDrInfo[i]["versionDr"] + "/" + "Support_Docs"
                        destination = newDrInfo[i]["labDr"]
                        moveDr(origin,destination)
        else:
                print("something␣went␣wrong.␣Exiting...")
                exit()

print("...and␣then␣there␣will␣be␣cake")
```