Санкт-Петербургский Национальный Исследовательский Университет

Информационных Технологий, Механики и Оптики

МФКТиУ, кафедра ПИиКТ

Лабораторная работа №4 по "Информатике"

Исследование языков разметки документов.

Выполнил: Анищенко Анатолий

Группа: Р3112

Вариант: 2

Санкт-Петербург

2018 г.

**Цель.**

Овладеть знаниями о различных современных языках разметки документов и форматах данных, навыками обработки данных с помощью языка Python 3.x.

**Задание.**

* Исходя из структуры расписания конкретного дня сформировать файл с расписанием в формате, указанном в задании в качестве исходного.
* Написать программу на языке Python, которая бы осуществляла парсинг и конвертацию исходного файла в новый

**Выполнение.**

**Schedule.json**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72 | {  "schedule" : {  "monday" : {  "class1" : {  "?today\_day" : "Пн",  "time" : {  "lessonTime" : {  "start": "08:20",  "end": "09:50"  },  "?parity" : "",  "?room" : "",  "location" : "ул.Ломоносова, д.9, лит. Е"  },  "room" : {  "roomNumber" : "3207 ауд.",  "location" : "ул.Ломоносова, д.9, лит. Е"  },  "lesson" : {  "lessonName" : "Иностранный язык в профессиональной деятельности (Прак)",  "?parity" : "",  "?teacher" : "",  "?href" : ""  }  },  "class2" : {  "?today\_day" : "",  "time" : {  "lessonTime" : {  "start": "10:00",  "end": "11:30"  },  "?parity" : "odd",  "?room" : "2434 ауд.",  "location" : "ул.Ломоносова, д.9, лит. Е"  },  "room" : {  "roomNumber" : "2434 ауд.",  "location" : "ул.Ломоносова, д.9, лит. Е"  },  "lesson" : {  "lessonName" : "Физика (Лаб)",  "?parity" : "odd",  "?teacher" : "Чистяков Виктор Владимирович",  "?href" : ""  }  },  "class3" : {  "?today\_day" : "",  "time" : {  "lessonTime" : {  "start": "10:00",  "end": "11:30"  },  "?parity" : "even",  "?room" : "2407 ауд.",  "location" : "ул.Ломоносова, д.9, лит. Е"  },  "room" : {  "roomNumber" : "2407 ауд.",  "location" : "ул.Ломоносова, д.9, лит. Е"  },  "lesson" : {  "lessonName" : "Физика (Прак)",  "?parity" : "even",  "?teacher" : "Чистяков Виктор Владимирович",  "?href" : ""  }  }  }  }  } |

**Schedule.proto**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123 | **message** **text** {  **message** **SCHEDULE** {  **message** **MONDAY** {  **message** **CLASS1** {  **optional** **string** today\_day = **0**  **message** **TIME** {  **message** **LESSONTIME** {  **required** **string** start = **0**  **required** **string** end = **1**  }  **required** LESSONTIME lessonTime = **0**  **optional** **string** parity = **1**  **optional** **string** room = **2**  **required** **string** location = **3**  }  **required** TIME time = **1**  **message** **ROOM** {  **required** **string** roomNumber = **0**  **required** **string** location = **1**  }  **required** ROOM room = **2**  **message** **LESSON** {  **required** **string** lessonName = **0**  **optional** **string** parity = **1**  **optional** **string** teacher = **2**  **optional** **string** href = **3**  }  **required** LESSON lesson = **3**  }  **required** CLASS1 class1 = **0**  **message** **CLASS2** {  **optional** **string** today\_day = **0**  **message** **TIME** {  **message** **LESSONTIME** {  **required** **string** start = **0**  **required** **string** end = **1**  }  **required** LESSONTIME lessonTime = **0**  **optional** **string** parity = **1**  **optional** **string** room = **2**  **required** **string** location = **3**  }  **required** TIME time = **1**  **message** **ROOM** {  **required** **string** roomNumber = **0**  **required** **string** location = **1**  }  **required** ROOM room = **2**  **message** **LESSON** {  **required** **string** lessonName = **0**  **optional** **string** parity = **1**  **optional** **string** teacher = **2**  **optional** **string** href = **3**  }  **required** LESSON lesson = **3**  }  **required** CLASS2 class2 = **1**  **message** **CLASS3** {  **optional** **string** today\_day = **0**  **message** **TIME** {  **message** **LESSONTIME** {  **required** **string** start = **0**  **required** **string** end = **1**  }  **required** LESSONTIME lessonTime = **0**  **optional** **string** parity = **1**  **optional** **string** room = **2**  **required** **string** location = **3**  }  **required** TIME time = **1**  **message** **ROOM** {  **required** **string** roomNumber = **0**  **required** **string** location = **1**  }  **required** ROOM room = **2**  **message** **LESSON** {  **required** **string** lessonName = **0**  **optional** **string** parity = **1**  **optional** **string** teacher = **2**  **optional** **string** href = **3**  }  **required** LESSON lesson = **3**  }  **required** CLASS3 class3 = **2**  }  **required** MONDAY monday = **0**  }  **required** SCHEDULE schedule = **0**  } |

**Lab4.py**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210  211 | **import** **re**  fin = open("Schedule.json", "r")  fout = open("Schedule.proto", "w")  **def** **getText**(arr):  text = ""  **for** i **in** range(len(arr)):  text += arr[i].rstrip().lstrip()  **return** text  **def** **sequence**(\*funcs):  **if** len(funcs) == **0**:  **def** **result**(src):  **yield** (), src  **return** result  **def** **result**(src):  **for** arg1, src **in** funcs[**0**](src):  **for** others, src **in** sequence(\*funcs[**1**:])(src):  **yield** (arg1,) + others, src  **return** result  numberRegex = re.compile(r"(-?(?:0|[1-9]\d\*)(?:\.\d+)?(?:[eE][+-]?\d+)?)\s\*(.\*)", re.DOTALL)  **def** **parseNumber**(src):  match = numberRegex.match(src)  **if** match **is** **not** **None**:  number, src = match.groups()  **yield** eval(number), src  stringRegex = re.compile(r'''("(?:[^\\']|\\['\\/bfnrt]|\\u[0-9a-fA-F]{4})\*?")\s\*(.\*)''', re.DOTALL)  **def** **parseString**(src):  match = stringRegex.match(src)  **if** match **is** **not** **None**:  string, src = match.groups()  **yield** eval(string), src  **def** **parseWord**(word, value=**None**):  l = len(word)  **def** **result**(src):  **if** src.startswith(word):  **yield** value, src[l:].lstrip()  result.\_\_name\_\_ = "parse\_%s" % word  **return** result  parseTrue = parseWord("true", **True**)  parseFalse = parseWord("false", **False**)  parseNull = parseWord("null", **None**)  **def** **parseValue**(src):  **for** match **in** parseString(src):  **yield** match  **return**  **for** match **in** parseNumber(src):  **yield** match  **return**  **for** match **in** parse\_array(src):  **yield** match  **return**  **for** match **in** parseObject(src):  **yield** match  **return**  **for** match **in** parseTrue(src):  **yield** match  **return**  **for** match **in** parseFalse(src):  **yield** match  **return**  **for** match **in** parseNull(src):  **yield** match  **return**  parseLeftSquareBracket = parseWord("[")  parsRightSquareBracket = parseWord("]")  parseEmptyArray = sequence(parseLeftSquareBracket, parsRightSquareBracket)  **def** **parse\_array**(src):  **for** \_, src **in** parseEmptyArray(src):  **yield** [], src  **return**  **for** (\_, items, \_), src **in** sequence(  parseLeftSquareBracket,  parseCommaSeparatedValues,  parsRightSquareBracket,  )(src):  **yield** items, src  parseComma = parseWord(",")  **def** **parseCommaSeparatedValues**(src):  **for** (value, \_, values), src **in** sequence(  parseValue,  parseComma,  parseCommaSeparatedValues  )(src):  **yield** [value] + values, src  **return**  **for** value, src **in** parseValue(src):  **yield** [value], src  parseLeftCurlyBracket = parseWord("{")  parseRightCurlyBracket = parseWord("}")  parseEmptyObject = sequence(parseLeftCurlyBracket, parseRightCurlyBracket)  **def** **parseObject**(src):  **for** \_, src **in** parseEmptyObject(src):  **yield** {}, src  **return**  **for** (\_, items, \_), src **in** sequence(  parseLeftCurlyBracket,  parseCommaSeparatedKeyvalues,  parseRightCurlyBracket,  )(src):  **yield** items, src  parseColon = parseWord(":")  **def** **parseKeyValue**(src):  **for** (key, \_, value), src **in** sequence(  parseString,  parseColon,  parseValue  )(src):  **yield** {key: value}, src  **def** **parseCommaSeparatedKeyvalues**(src):  **for** (keyvalue, \_, keyvalues), src **in** sequence(  parseKeyValue,  parseComma,  parseCommaSeparatedKeyvalues,  )(src):  keyvalue.update(keyvalues)  **yield** keyvalue, src  **return**  **for** keyvalue, src **in** parseKeyValue(src):  **yield** keyvalue, src  **def** **parse**(s):  s = s.strip()  match = list(parseValue(s))  **if** len(match) != **1**:  **raise** **ValueError**("not a valid JSON string")  result, src = match[**0**]  **if** src.strip():  **raise** **ValueError**("not a valid JSON string")  **return** result  **def** **jsonToProtobuf**(jsonObj, name="text"):  result = "message " + name + " {"  result += processObject(jsonObj)  **return** result    **def** **processObject**(jsonObj, linePadding=""):  result = ""  index = **0**  **for** key **in** jsonObj:  itemType = type(jsonObj[key])  **if** itemType **is** dict **or** itemType **is** list:  result += addObjectsArrayMessage(key, jsonObj[key], index, linePadding + "**\t**")  **else**:  result += getKeyLabels(key, jsonObj[key], index, linePadding + "**\t**")  index += **1**  **return** result + "**\n**" + linePadding + "}**\n**"  **def** **addObjectsArrayMessage**(item, jsonObj, index, linePadding):  **if** type(jsonObj) **is** list:  **return** addArrayMessage(item, jsonObj, index, linePadding)  **else**:  **return** addObjectMessage(item, jsonObj, linePadding) + getKeyLabels(item, jsonObj, index, linePadding)  **def** **addObjectMessage**(key, value, linePadding):  key = key[**1**::] **if** key[**0**] == '?' **else** key  newMessage = "**\n\n**" + linePadding +"message " + key.upper() + " {" + processObject(value, linePadding)  **return** newMessage    **def** **addArrayMessage**(key, value, index, linePadding):  objectsType = type(value[**0**])  sameTypeArray = **True**  **for** item **in** value:  sameTypeArray = sameTypeArray **and** (type(item) == objectsType)  **if** sameTypeArray:  **if** objectsType **is** **not** list **and** objectsType **is** **not** dict:  **return** "**\n**" + linePadding + "repeated" + ("float" **if** objectsType **is** int **or** objectsType **is** float **else** "bool" **if** objectsType == bool **else** "string") + " " + key + " = " + index  **else**:  **return** addObjectMessage(key, value[**0**], linePadding) + "**\n**" + linePadding + "repeated " + key.upper() + " " + key + " = " + str(index)  **else**:  **return** "**\n**Error - Cannot parse. Array is having different typed elements."    **def** **getKeyLabels**(item, value, index, linePadding):  variableType = "optional" **if** item[**0**] == '?' **else** "required"  item = item[**1**::] **if** item[**0**] == '?' **else** item  **if** type(value) **is** dict **or** type(value) **is** list:  **return** "**\n**" + linePadding + variableType + " " + item.upper() + " " + item + " = " + str(index)  **else**:  **return** "**\n**" + linePadding + variableType + " "+ ("string" **if** type(value) **is** str **else** "double" **if** type(value) **is** int **or** type(value) **is** float **else** "bool") + " " + item + " = " + str(index)    text = getText(fin.readlines())  shedule = parse(text)  print(jsonToProtobuf(shedule))  print(jsonToProtobuf(shedule), file=fout)  fin.close()  fout.close() |
|  |  |

**Выводы.**

Я овладел знаниями о различных современных языках разметки документов и форматах данных, навыками обработки данных с помощью языка Python 3.x.