XML-Ը JSON-Ի ՁԵՎԱՓՈԽՈՂ ԾՐԱԳՐԱՅԻՆ ՄՈԴԵԼԻ ԱՇԽԱՏԱՆՔԻ ՍԿԶԲՈՒՆՔԸ

Գևորգ Իվանի Մարգարով
պ.գ.թ., պրոֆեսոր
ՀԱՊՀ, ՏՀՏԷ ինստիտուտ, ՏԱԾԱ ամբիոն
gmargarov@gmail.com
Նարեկ Լևոնի Նալթակյան
մագիստրանտ
ՀԱՊՀ, ՏՀՏԷ ինստիտուտ, ՏԱԾԱ ամբիոն
nareknaltakyan1@gmail.com
Աղասի Թադևոսի Սեյրանյան
մագիստրանտ
ՀԱՊՀ, ՏՀՏԷ ինստիտուտ, ՏԱԾԱ ամբիոն
aghasi.seyranyan@gmail.com
Մարտին Աշոտի Միրզոյան
մագիստրանտ
Չինաստանի էլեկտրոնային գիտության և տեխնոլոգիայի համալսարան,
Համակարգչային գիտության և ճարտարագիտության դպրոց
martinmirzoyan00@gmail.com

Ամփոփագիր

Ներկա ժամանակներում ծրագրային համակարգերի աշխատանքի մեջ հաճախ օգտագործվում են տարբեր տիպերի պատկանող ֆայլեր, որոնք ժամանակ առ ժամանակ կարիք են ունենում փոփոխության, թե ծրագրային տեղեկության տեսանկյունից և թե տվյալ ֆայլի տիպի տեսանկյունից:

Բազմաթիվ համակարգերի համար խնդիր է հանդիսանում օգտագործվող ֆայլերի մի տեսակից մյուսին անցնելը, որի համար մշակվել է ծրագրային մոդել՝ միկրոսերվիսի տեսքով, որը հնարավորություն է տալիս առանց խնդիրների կառավարել այդ անցումը՝ հաշվի առնելով բոլոր հնարավոր բացառիկ իրավիճակները: Ծրագրի մոդելի մշակման համար ընտրվել է GoLang ծրագրավորման լեզուն: Հիմնականում նմանատիպ խնդիր կարող է առաջանալ դատական, բանկային և այլ ոլորտների համար նախատեսված համակարգերում, որոնք գրվել են դեռ նախորդ դարում, երբ արդիական են եղել XML մոդելի ֆայլերը և նրանց աշխատանքի հիմքում դրվել են հենց այդ ֆայլերը: Սակայն, հետագայում ստեղծվել են այնպիսի ֆայլերի մոդելներ, որոնք բազմաթիվ հատկանիշներով գերազանցում են XML մոդելի ֆայլերին:

Այս հոդվածում քննարկվում է մի նմանատիպ դեպք, որտեղ դրված է խնդիր XML մոդելի ֆայլը անհրաժեշտաբար ձևափոխելու JSON մոդելի ֆայլի՝ առանց ծրագրային տեղեկության կորստի:

Հիմնաբառեր. XML, JSON, միկրոսերվիս, GoLang, ֆայլային համակարգ:

Ներածություն

Վերջին ժամանակներում տեխնոլոգիաները զարգանում են անհամեմատելիորեն արագ, և արդյունքում բազմաթիվ տեխնոլոգիաներ, մեթոդներ, ալգորիթմներ կորցնում են իրենց արդիականությունը: Շատ համակարգեր, որոնք չեն ենթարկվում թարմացումների և շարունակում են կիրառել ոչ արդի լուծումներ, հետագայում ունենում են մի շարք խնդիրներ: Այդպիսի խնդիրների շարքին են դասվում ֆայլային համակարգի հիմքում ընկած ֆայլերի մոդելների տեսակների փոփոխությունը: Հիմնականում նմանատիպ խնդիր կարող է առաջանալ դատական, բանկային և այլ ոլորտների համար նախատեսված համակարգերում, որոնք գրվել են դեռ նախորդ դարում, երբ արդիական են եղել XML մոդելի ֆայլերը և նրանց աշխատանքի հիմքում դրվել են հենց այդ ֆայլերը: Սակայն հետագայում ստեղծվել են այնպիսի ֆայլերի մոդելներ, որոնք բազմաթիվ հատկանիշներով գերազանցում են XML-ին: Այսպիսի խնդիրների լուծման համար կա երկու ճանապարհ՝ կա՛մ համակարգի ամբողջական վերաշարադրում, կա՛մ մոդիֆիկացիա, որը ենթադրում է տվյալ համակարգի ինչ-որ ֆունկցիոնալ հատվածի փոփոխություն կամ ընդլայնում: Եթե համակարգը չափերով մեծ է կամ աշխատում է այնպիսի ոլորտում, որտեղ նրա կասեցումն անընդունելի է, ապա դրա վերաշարադրման դեպքում առաջացած ծախսերն ու խնդիրները կլինեն ավելի շատ, քան նրանից ստացված արդյունքը:

Այսպիսով՝ երկրորդ ճանապարհը խելամիտ լուծում կարող է լինել այս խնդրի համար. դրա միջոցով իրականացվում է տվյալ համակարգի ֆունկցիոնալի մոդիֆիկացում: Սակայն, պետք է նշել, որ այս դեպքում ևս կան խնդիրներ՝ կախված այն հանգամանքից, թե այդ համակարգի ծրագրային ճարտարապետությունը ինչ մոդելով է գրվել: Այսպիսի խնդիրներից խուսափելու կամ դրանց ավելի արդյունավետ լուծում տալու համար հիմնականում օգտագործվում է միկրոսերվիսային ճարտարապետական մոդելը, որը ենթադրում է նմանատիպ ֆունկցիոնալների համախմբման և առանձին համակարգի մեջ տեղադրելու գաղափարը, այսինքն՝ ամբողջական համակարգը բաղկացած է լինում միմյանցից տարբեր միկրոսերվիսներից, որոնցից յուրաքանչյուրը պարունակում է ֆունկցիոնալ տեսանկյունից միանման կոդի հատվածներ:

Խնդրի դրվածքը և մեթոդիկայի հիմնավորումը

Խնդրի դրվածքը հետևյալն է՝ գոյություն ունի մի համակարգ, որի ֆայլային համակարգի հիմքում ընկած են XML մոդելի ֆայլերը, որոնք պետք է ձևափոխել JSON-ի:

JSON-ի առավելությունները XML-ի նկատմամբ հետևյալն են.

- JSON մոդելում, ի տարբերություն XML-ի, առկա են տվյալների տարբեր տիպեր՝ տողային, թվային, զանգված, բուլյան, ինչը ենթադրում է, որ ծրագրավորողների կողմից սխալ թույլ տալու հավանականությունն ավելի փոքր կլինի: Իսկ XML-ում գրված ցանկացած ինֆորմացիա դիտարկվում է որպես string տիպի,
- JSON-ը աջակցվում է գրեթե բոլոր բրաուզերների կողմից, իսկ XML-ի դեպքում մի բրաուզերից մյուսին անցնելիս կարող են խնդիրներ առաջանալ,
- JSON-ը, ի տարբերություն XML-ի, ավելի քիչ տարածք է զբաղեցնում հիշողության մեջ, օրինակ՝

JSON:

```
{"books":[
```

```
{ "author":"Հովհաննես Թումանյան",
"name":"Քաջ Նազարը" },
{ "author":"Շիրվանզադե",
"name":"Արտիստը" },
{ "author":"Եղիշե Չարենց",
"name":"Հեռացումի խոսքեր" }
]}
```

XML:

```
<books>
<book>
<author>Հովհաննես
Թումանյան</author>
<name>Քաջ Նազարը</name>
</book>
<book>
<author>Շիրվանզադե</author>
<name>Արտիստը</name>
</book>
<book>
<author>Եղիշե Չարենց</author>
<name>Հեռացումի
խոսքեր</name>
</book>
</books>
```

- Օգտագործողի տեսանկյունից JSON-ը ավելի ընթեռնելի է, քան՝ XML-ը,
- XML-ը բացի բուն ինֆորմացիայից պարունակում է նաև ավելորդ տողեր,
- JSON-ը աջակցվում է AJAX-ի ավելի շատ գործիքների կողմից, քան՝ XML-ը,
- JSON-ի դեպքում JavaScript-ի ձևափոխելու համար կարելի է օգտագործել JavaScript-ի eval() ֆունկցիան, իսկ XML-ի դեպքում պետք է հավելյալ գործողություններ կատարել,
- JSON-ում հնարավոր է օգտագործել զանգվածներ, իսկ XML-ում՝ ոչ [Walker]:

Այսպիսով, որպեսզի համակարգը փոփոխենք, անհրաժեշտ է այն կա՛մ ամբողջությամբ վերաշարադրել, կա՛մ դրա ինչ-որ ֆունկցիոնալ հատված մոդիֆիկացնել: Համակարգի վերաշարադրումն արդյունավետ գործնթաց չէ, քանի որ պահանջում է մեծաքանակ ռեսուրսներ կամ կարող է աշխատանքի կասեցման պատճառով առաջացնել խնդիրներ:

Հավելյալ խնդիրներից խուսափելու համար պետք օգտագործել այնպիսի ծրագրային ճարտարապետական մոդել, որը բաց կլինի փոփոխությունների համար և որում փոփոխություններ մտցնելն անդառնալի հետևանքներ չի ունենա, որպես այդպիսի մոդել կարող է ծառայել ներկայումս մեծ հայտնիություն վայելող միկրոսերվիսային ճարտարապետական մոդելը: Ի տարբերություն մոնոլիտ ճարտարապետական մոդելի՝ միկրոսերվիսային մոդելն ունի մի շարք առավելություններ: Դրանք են՝

- Հեշտ է ընկալման տեսանկյունից: Միկրոսերվիսային ծրագրի բաժանումը փոքր և պարզ բաղադրիչների ավելի ընթեռնելի է դարձնում հավելվածը:
- Ընդարձակելիություն: Միկրոսերվիսային հավելվածների ևս մեկ առավելությունն այն է, որ ամեն մի ծառայություն կարող է ընդարձակվել մյուսներից անկախ: Այսպիսով՝ վերջինիս գործընթացի վրա ավելի քիչ ժամանակ և ռեսուրսներ են ծախսվում, քան կծախսվեին, եթե հավելվածը լիներ մոնոլիտ ճարտարապետությամբ:
- Տեխնոլոգիայի ընտրության ճկունություն: Կարելի է չսահմանափակվել տեխնոլոգիաների ընտրության հարցում: Կարող է յուրաքանչյուր միկրոծառայության համար ընտրվել նոր տեխնոլոգիա:
- Ճկունության ավելի բարձր մակարդակ: Հավելվածի միկրոծառայության կամայական սխալն անդրադառնում է

միայն այդ կոնկրետ ծառայության վրա, այլ ոչ թե՝ ողջ ծրագրի: Այսպիսով, նոր ծառայությունների ավելացումը կամ թարմացումը ավելի քիչ ռիսկեր են պարունակում:

- Անկախ կոմպոնենտներ: Բոլոր ծառայությունները կարող են փոփոխվել կամ թարմացվել իրարից անկախ՝ ինչը մեծ ճկունության հնարավորություն է տալիս: Մեկ միկրոծառայության սխալն ազդում է միայն այդ ծառայության վրա և չի ազդում բուն ծրագրի աշխատանքի վրա: Բացի այդ, ի տարբերություն մոնոլիտ ճարտարապետությունների, միկրոսերվիսային ճարտարապետությամբ ստեղծված ծրագրում ավելի հեշտ է ավելացնել նոր ծառայություններ [Gnatyk]:

Նմանատիպ համակարգեր նախագծելու համար նախընտրելի է օգտագործել GoLang ծրագրավորման լեզուն: Go- ն բաց աղբյուրով ծրագրավորման լեզու է, որը հեշտացնում է պարզ, հուսալի և արդյունավետ ծրագրակազմի(software) կառուցումը: GoLang-ը ունի բազմաթիվ առավելություններ, դրանցից են՝

- Տիպի անվտանգություն. սխալ տիպերի օգտագործման հետ կապված սխալները ազդանշվում են ոչ թե կատարման, ալ կազմման ընթացքում՝ այդպիսով հնարավորություն տալով խուսափել հետագայում առաջացող բազմաթիվ խնդիրներից:
- Go-ն, ի տարբերություն ներկայիս տարածված ծրագրավորման լեզուների, ունի բարձր արտադրողականություն, դրա միջոցով գրված կոդը ընթեռնելի է և հակիրճ:
- Ունի մեծ ֆունկցիոնալություն ֆայլերի հետ աշխատելու համար [Effective Go]:

Ալգորիթմի նկարագրություն

Ալգորիթմի աշխատանքը բաժանվում է երկու մասի՝ առաջին հատվածը, որը պատասխանատու է XML-ը JSON-ի ձևափոխելու համար, և երկրորդ հատվածը, որը նախատեսված է արդեն պատրաստի JSON-ի մեջ փոփոխություններ մտցնելու համար:

```
func main() {
 err:= convert.ParseXmlToJson(xmlFilePath,
jsonFilePath, xmlFileType)
 if err != nil {
 fmt.Println(err)
 }
 err = convert.JsonChanges(...)
 if err != nil {
 fmt.Println(err)
 }
}
```

convert.ParseXmlToJson մեթոդը որպես պարամետր ստանում է երեք փոփոխական՝

1. Xml ֆայլի տեղակայման ճանապարհը,
2. Հետագայում ստեղծվող JSON ֆայլի տեղը և անվանումը,
3. Xml ֆայլի տեսակը, քանի որ Xml ֆայլերը իրենց հերթին լինում են մի քանի տեսակի:

Իսկ convert.JsonChanges-ի իմպլեմենտացիան հոդվածում գրված չէ, քանի որ նախատեսված է, որ տվյալ համակարգի ծրագրավորողները ցանկության դեպքում JSON-ի մեջ կարող են փոփոխություններ մտցնել:

```
func ParseXmlToJson(xmlFilePath string,
jsonFilePath string, xmlFileType string) error {
 xsltStr, root:= getXslt(xmlFileType) // տվյալների բազայից վերցվում են տվյալ ֆայլին հատուկ xslt-ն և նրան հատուկ արմատային թեգը
 xmlFile, err:= os.Open(xmlFilePath) // բացվում է XML ֆայլը
 if err != nil {
 return errors.New("Opening file error: " +
err.Error())
 }
```

```
 defer xmlFile.Close() //Մեթոդի աշխատանքի վերջում փակվում է XML ֆայլը
 xmlData, err:= ioutil.ReadAll(xmlFile)
 if err != nil {
 return errors.New("Reading file error: " + err.Error())
 }
 xmlData = replaceAll160SpaceTo32Space(xmlData) //փոխարինումը 160 (Non Breaking Space) դեպի Char 32(Normal Space)
```

Այս հատվածում կարելի է xml ինֆորմացիայի հետ փոփոխություններ կատարել, օրինակ՝ եթե թեգեր կան, որոնք այլ տեսք պիտի ունենան կամ այլ անվանում:

```
      xmlDataOfTheRoot, err:= cutDataOfTheRoot(xmlData, root) //Քանի որ xml-ը ավելորդ տողեր է պարունակում հեռացվում են այդ տողերը և մնում է բուն ինֆորմացիան
 if err != nil {
 return errors.New("Xml Cutting of the root error: " + err.Error())
 }
 encodedData, err = replaceBracketToText(xmlDataOfTheRoot) //Քանի որ փակագծերը({}) xml-ը json-ի ձևափոխելուց կարող են առաջացնել խնդիրներ, ապա նրանք փոխարինվում են տողերով
 mapData, err:= mxj.NewMapXml([]byte(encodedData)) //ձևափոխում է xml ինֆորմացիան տվյալների կառուցվածքներից քարտեզով
 if err != nil {
 return errors.New("Xml Parsing error: " + err.Error())
 }
 jsonData, err:= mapData.Json() //Ձևափոխում է քարտեզը JSON տիպի ինֆորմացիայի
 if err != nil {
 return errors.New("Json Parsing error: " + err.Error())
 }
 jsonFile, err:= os.Create(jsonFilePath) //Ստեղծում և բացում է JSON տիպի ֆայլ
 if err != nil {
 return err
 }
 defer jsonFile.Close() //Մեթոդի աշխատանքի վերջում ֆայլը փակելու համար է նախատեսված
 if _, err:= jsonFile.Write([]byte(jsonData));
 //Ինֆորմացիան արտագրում է JSON տիպի ֆայլում
 err != nil
 return errors.New("Json writing to file error: " + err.Error())
 }
 return nil
}
```

Հետազոտության արդյունքները

Տվյալ համակարգը կարելի է օգտագործել կառավարական համակարգերում: Օրինակ՝ Եվրոպայի պառլամենտի իրավական համակարգերի ֆայլերը հիմնականում լինում են xml տիպի, քանի որ գրվել են նախորդ դարում, և այդ իսկ պատճառով այդպիսի համակարգերի ֆայլերը զբաղեցնում են անհամեմատ մեծ տարածք, օրինակ՝ 02013R0575-20191225_EN ֆայլի Xml մոդելը զբաղեցնում է 2,3ՄԲ տարածք, իսկ JSON մոդելը 1,9ՄԲ, այսինքն՝ գործնականում ֆայլի չափերը փոքրացել են 0,4MB-ով (նկ.1), 02014R0600-20160701_EN՝ ֆայլի չափերը փոքրացել են 53KB-ով (նկ.2):

Նկ. 1.

02013R0575-20191225_EN ֆայլի ձևափոխման արդյունքը.

Նկ. 2.

02014R0600-20160701_EN EN ֆայլի ձևափոխման արդյունքը.

Եզրակացություն

Ներկայումս աշխատող տեխնոլոգիապես հին համակարգերի, և ինչու ոչ, նաև որոշ նոր համակարգերի խնդիրը նրանց մոդիֆիկացիայի համար փակ լինելու մեջ է: Համակարգի ճարտարապետության հիմքում ճիշտ մոդելների ընտրությունը բերում է վերջինիս անվտանգության և արագագործության մեծացման, ինչի արդյունքում համակարգը միևնույն ժամանակում ավելի շատ գործողություններ է կարողանում կատարել: Ցանկացած համակարգ կարելի է փոփոխության ենթարկել այնքանով, որքանով մեզ թույլ կտան այդ համակարգի ֆունկցիոնալ պահանջները, սակայն միշտ չէ, որ փոփոխություն կատարելը ճիշտ որոշում է: Երբեմն ավելի ճիշտ է վերաշարադրել տվյալ համակարգը, քան այն փոփոխել:

Գրականության ցանկ

1. Alysa Walker, “JSON vs XML: What's the Difference?”: Guru99, 01. September 2022, https://www.guru99.com/json-vs-xml-difference.html
2. Romana Gnatyk, “Microservices vs Monolith: which architecture is the best choice for your business?”: N-iX, 01. September 2022, https://www.n-ix.com/microservices-vs-monolith-which-architecture-best-choice-your-business/
3. “Effective Go”, 01. September 2022, https://golang.org/doc/effective_go

THE OPERATION PRINCIPLE OF A SOFTWARE MODEL TRANSFORMING XML TO JSON

Gevorg Margarov
PhD in Technical Sciences, Professor
NPUA, ITTE Institute, ISSD Chair
gmargarov@gmail.com
Narek Naltakyan
MA Student
NPUA, ITTE Institute, ISSD Chair
nareknaltakyan1@gmail.com
Aghasi Seyranyan
MA Student
NPUA, ITTE Institute, ISSD Chair
aghasi.seyranyan@gmail.com
Martin Mirzoyan
MA Student
UESTC, CSE School
martinmirzoyan00@gmail.com

Abstract

Nowadays, in the operations of software systems, files belonging to different types are often used, which from time to time need to be changed, both from the point of view of the program information and from the point of view of the given file type.

For many systems, it is a problem to switch from one type of a file to another, and for them a software model has been developed in the form of a microservice, which makes it possible to manage this transition without problems, considering all possible exceptional situations. The GoLang programming language was chosen for the development of the project model. Basically, a similar problem can arise in systems designed for judicial, banking, and other fields, which were written in the last century, when XML model files were up-to-date, and these files were the basis of their work. However, in the future, such file models were created, which are superior to the XML model files in many features. This article discusses a similar case, where the problem is that an XML model file needs to be converted into a JSON model file without losing program information.

Keywords: XML, JSON, microservice, GoLang, file system.

Ներկայացվել է՝ 07.10.2022թ.
Ուղարկվել է գրախոսման՝ 01.11.2022թ.