행운의 룰렛(stw)이벤트와 댄싱9을 작업하면서 사용한 다국어 처리 방법 3--5개에 대해 정리해 보았습니다. 가장 단순한 방법부터 좀 여러 위치를 손대야 하는 방법 순서대로 설명합니다. 제일 밑에 여러 언어 국가 코드를 설명한 부분도 있습니다.

다국어 - STW를 예로

FTL에서

if elseif 구문 사용이 가장 간단하다. ftl문법으로 보면 다음과 같다. lang은 mwave에서 언제나 ftl로 부터 접근 가능하다. 현재 mwave에서는 lang 변수에 KOR, JPN, CHN, TWN, ESP중 하나가 담긴다. 이벤트에서만 쓴 적이 있다. 급할 때는 써도 좋다.

행운의 룰렛 이벤트에서 타이틀 부분은 아래처럼 만들었다.

Listing 1: if elseif

```
<title>
    <#if lang=="KOR">
        Mwave 행운의 룰렛
    <#elseif lang == "JPN">
        Mwave幸運のルーレット
    <#elseif lang == "CHN">
        Mwave幸运转盘
    <#elseif lang == "TWN">
        Mwave幸運轉盤
    <#elseif lang == "ESP">
        Mwave's Spin the Wheel
    <#else>
        Mwave's Spin the Wheel
    </#if>
</title>
```

---

ftl전체가 너무 길어지기에 각각의 번역을 함수로 하나씩 만들고 translation.ftl 파일에 묶을 수 있다.

Listing 2: translation.ftl

```
<#function stw_title>
  <#if lang=="KOR">
    <#return "Mwave 행운의 룰렛">
  <#elseif lang == "JPN">
    <#return "Mwave幸運のルーレット">
    ...
    ...
  </#if>
</#function>
```

그리고 이 translation.ftl을 import 한 후에

import 구문을 아래와 같이 위에 두고

```
<#import "/spinthewheel/translation.ftl" as t />
```

`t.stw_title()`처럼 호출한다.

```
<title>${t.stw_title()}</title>
```

함수를 여러개 만들 때 손이 많이 간다. 해서 번역하는 분께 excel파일을 받았다면 excel파일을 csv로 바꾼 다음 자동으로 같은 ftl을 만들어 주는 스크립트를 하나 작성했다. csv에 keyword(stw_title과 같은)를 붙인 다음 그 keyword 이름을 붙이고 mwave에서는 `src\main\resources\translation\gen_translation_ftl.rb`을 실행해 translation.ftl을 생성했다.

stw.csv의 내용이 다음과 같으면 위와 동일한 ftl이 생성된다.

Listing 3: stw.csv

```
"stw.title","Mwave 행운의룰렛 ","Mwave's Spin the Wheel","Mwave 幸運の
    ルーレット","Mwave 幸运转盘","Mwave 幸運轉盤",
```

mwave에서 사용하는 gen_translation_ftl.rb은 jruby로 구현한 코드이고 [1] csv를 읽어 ftl을 생성한다. csv에 순서대로 ko_KR, en_US, ja_JP, zh_CN, zh_TW, es_ES의 번역을 넣어주어야 한다.

[1] X로 만드는 것이 더 좋지 않을까요? 예 그것도 괜찮습니다.

Listing 4: gen_translation_ftl.rb

```
# vim: fileencoding=utf-8 :
SPACE="\u0020"

require 'csv'
require 'date'

File.open('../../webapp/front/ftl/promotion/spinthewheel
    /translation.ftl', 'w:UTF-8') do |f|

  CSV.foreach('stw.csv') do |row|
    row = row.collect do |e|
      if e then
        e.force_encoding('utf-8')
      end
    end

    t={}
    key, t['ko_KR'], t['en_US'], t['ja_JP'], t['zh_CN'], t['zh_TW
        '], t['es_ES'] = row.collect do |e|
      if e then
        e.strip
      end
    end
    if !key then
      next
    end

    t.keys.each do |langType|
      if not t[langType] then
        t[langType] = t['en_US']
      end
      puts langType, t[langType]
    end

    param = ''
    ftl_replace = ''
    if /\{0\}/ =~ t['ko_KR'] then
      param = SPACE+'arg'
      ftl_replace = '?replace("{0}", arg)'
    end

    macro =<<EOF
<#function #{key.sub('.', '_')}#{param}>
  <#if lang=="KOR">
    <#return "#{t['ko_KR']}"#{ftl_replace}>
  <#elseif lang == "JPN">
    <#return "#{t['ja_JP']}"#{ftl_replace}>
  <#elseif lang == "CHN">
    <#return "#{t['zh_CN']}"#{ftl_replace}>
  <#elseif lang == "TWN">
    <#return "#{t['zh_TW']}"#{ftl_replace}>
  <#elseif lang == "ESP">
    <#return "#{t['es_ES']}"#{ftl_replace}>
  <#else>
    <#return "#{t['en_US']}"#{ftl_replace}>
  </#if>
```

```
</#function>

EOF
    puts macro
    f.write macro
  end
end
```

ftl단에서는 이렇게 번역을 처리할 수 있다. java단에서 번역이 필요할 때는 어떻게 할 것인가?

Java에서

특히 Spring에서 `MessageSource(org.springframework.context.MessageSource)`로 I18N을 처리하는 경우가 많고 BASE에서도 mwave에서도 MessageSource Interface를 사용한다. 이렇게 Controller에서 사용할 수 있다.

Listing 5: SpintTheWheelController.java

```
@Controller
public class SpinTheWheelController extends CommonController {
    ...
    @Autowired
    MessageSource messageSource;

    @RequestMapping(value = { "/promotion/spinthewheel
        /save_email" })
    public MJResponse some_action(@RequestParam String token) {
        MemberVO memberVO = MemberVO.getLogined(request);
        if (memberVO == null || !isEmailAccount(memberVO)) {
            return ResponseBuilder.failure(_T(request, "stw.
                email_id_login_needed"));
        }

        ...

        return ResponseBuilder.success();
    }

    private String _T(HttpServletRequest request, String code) {
        return messageSource.getMessage(code, null,
            localeFromRequest(request));
    }

    private Locale localeFromRequest(HttpServletRequest request) {
        String lc = (String) request.getAttribute("languageCountry")
            ;
        String[] arr = StringUtils.split(lc, "_");
        return new Locale(arr[0], arr[1]);
    }

    ...
}
```

번역을 얻기 위해 _T함수에서 getMessage함수를 사용했다.

```
messageSource.getMessage(code, null, locale)
```

code에 stw.title과 같은 임의로 정한 label을, locale에 java.util.Locale 객체를 넣는다. Locale에 언어와 국가 정보가 담겨있다. Mwave에서는 langaugeCountry라는 모든 페이지에서 접근가능한 값을 사용해 java.util.Locale 객체를 만들었다. mwave에서는 languageCountry라는 값을 cookie와 IP를 GeoIP에 넣어 받은 국가코드를 바탕으로 정한다. [2]

mwave에서 messageSource bean은 applicationContext.xml에 이렇게 설정한다. BASE프로젝트에서 만든 me.interest.util.DatabaseMessageSource를 사용하고 있다.

Listing 6: applicationContext.xml

```
<bean id="messageSource" class="me.interest.util.
    DatabaseMessageSource">
    <property name="sqlId" value="me.interest.util.
        selectInternationalization" />
    <property name="langKey" value="langKey" />
    <property name="langValue" value="langValue" />
</bean>
```

kproject-front의 경우에 ibatis의 me.interest.util.selectInternationalization를 통해 CT_I18N 테이블에서 읽어오도록 되어 있다.

CT_I18N 테이블에 값을 넣는 것은 INSERT문을 타이핑해서 DB에 넣어도 된다. 여기서는 gen_translation_ftl.rb 중간에 CT_I18N에 upsert(있으면 update, 없으면 insert하는) 코드를 넣었다. 해서 번역을 변경할 때 작업순서는 stw.csv를 고치고 gen_translation_ftl.rb를 실행했었다.

```
cti18nupdate.upsert({
  :langType => langType,
  :langKey => key,
  :langValue => t[langType]
})
```

csv를 고칠 때 excel로 고치면 DRM이 적용되어서 csv파일 자체의 내용이 변하기 때문에 OpenOffice Calc를 사용하였다.

댄싱9에서의 다국어 처리

messageSource는, 여기서 DatabaseMssageSource가 아니고 ResourceBundleMessageSource를 사용하고

```
<bean id="messageSource"
```

[2] com.mnet.kproject.system.InterceptorUtil.preLocation() 함수에 lang과 langCountry를 정하는 로직이 들어있다. mwave에는 struts를 사용한 코드와 spring을 사용한 코드가 같이 존재하는데 InterceptorUtil이 양쪽에 같이 사용된다.

```
      class="org.springframework.context.support.
          ResourceBundleMessageSource">
    <property name="basename" value="messages/messages"></property>
  </bean>
```

ftl에서는 이렇게 참조한다. (위에서 사용한 방법과 다르게, ftl안에서 messageSource를 쓰도록 했다) [3]

```
<@spring.messageText "n_vote", [vote_team_infos.red.voteCnt] />
```

인자가 하나라도 배열형태로 넘긴다.

messageSource는 messaages/message_{lang}.properties파일을 읽어서 사용한다. `src/main/resources`디렉토리에 5개의 property파일을 만들었다. message_en.properties, message_ja.properties, message_ko.properties, messages_zh_CN.properties, messages_zh_TW.properties 이고 각각의 내용은 아래와 같다. [4]

Listing 7: message$_{lang}.properties$

```
# en
n_rank={0,choice,0#zero|1#1st|2#2nd|3#3rd|3<{0}th}
n_vote={0} votes
n_age={0} years old

# ja
n_rank={0}位
n_vote={0}票
n_age={0}歳

# ko
n_rank={0}위
n_vote={0}표
n_age={0}세

# zh_CN
n_rank=第{0}名
n_vote={0}票
n_age={0}岁

# zh_TW
n_rank=第{0}名
n_vote={0}票
n_age={0}歲
```

여기서 en의 n_rank에서 사용한 `{0,choice,0#zero|1#1st|2#2nl|3#3rd|3<{0}th}` 구문은 java의 I18N에서 지원하는 문법이다.

한 단어나 한 문장 수준의 번역이 필요할 때 messageSource를 사용하거나 FTL 함수 호출로 처리하는 해도 좋다. 한데 한 페이지나 단락 범위의 번역이

[3] 재시작이 필요한가요? 그랬던 것으로 기억합니다. 확실치 않습니다.

[4] property를 이클립스에서 쓸 때는 무슨 plugin을 하나 설치해야 화면에 글자가 uac00처럼 표시되지 않고 편집하기도 편하다.

필요할 때는 keyword(혹은 label)을 다 만들어 주어야 하는 불편함이 있다. 해서 UI개발팀에서 받은 HTML이 있다면 그것으로 한글 ftl을 만든다. 그리고 번역이 담긴 csv를 준비하고 node별로 번역하는 스크립트들을 만들어 썼다. 예를들어 댄싱9의 MC소개 페이지의 한글 버전은 파일이름이 mcintro_con_body.kor.ftl 이고 내용이 다음과 같다.

Listing 8: mcintro_con_body.kor.ftl

```
<div class="con_body mcintroduction">
  <p class="photo"><img src="${img_asset_host}/images
      /pic_mc_photo.jpg" alt오상진=""></p>
    <div class="desc">
      <dl class="overview">
          <dt오상진></dt>
          <dd국내> 최초의댄스서바이벌프로그램    &lt댄싱;9&gt의; 만능 MC</dd>
      </dl>
      <ul class="filmography">
          <li전> 아나운서 </li>
          <li>MBC &lt불만제로;&gt;,&lt생방송; 화제집중&gt의; MC </li>
          <li>MBC &lt일밤;&gt;,&lt댄싱; 위드더스타  &gt; 외다수의예능프로그
              램진행및출연      </li>
      </ul>
    </div>
  </div>
</div>
```

이렇게 원본 *.kor.ftl을 마련해 두고 conv_translation.py 스크립트를 실행하면 translation.csv안의 이런 값을 읽어

Listing 9: translation.csv

```
"오상진","Oh Sang Jin","オ·サンジン","吴尚镇","吳尚鎮"
"국내 최초의 댄스 서바이벌 프로그램 <댄싱9>의 만능 MC","The MC of Korea's
    first Dance Survival Program, 'Dancing9'!","韓国初のダンスサバイバル
    番組Dancing9の万能MC","韩国最初的舞蹈选秀节目<dancing9>的万能主持人","韓國
    最初的舞蹈選秀節目<dancing9>的萬能主持人"
"전 아나운서 ","Announcer ","元アナウンサー","前任主持人","前任主持人"
"MBC <불만제로>,<생방송 화제집중>의
    MC ","MBC's 'Complaint Zero', 'On Focus' MC","MBC不満ゼロ生放送 話題
    集中のMC","曾担任MBC<不满zero>, <直播话题集中>的主持","曾擔任MBC<不滿
    zero>, <直播話題集中>的主持"
"MBC <일밤>,<댄싱 위드 더 스타> 외 다수의 예능 프로그램 진행 및 출
    연","Appeared on MBC's 'Sunday Night', 'Dancing with the Stars'
    and many other variety shows","MBC日晩ダンシング·ウィズ·ザ·
    スター他多数のバラエティ番組の司会および出演","主持或出演MBC<星期天晚上>, <
    Dancing With The Stars>等多数综艺节目","主持或出演MBC<星期天晚
    上>, <Dancing With The Stars>等多數綜藝節目"
```

각 HTML 노드안의 텍스트를 교체해서 mcintro.eng.ftl을 만든다. [5]

Listing 10: mcintro_con_body.eng.ftl

```
<div class="con_body mcintroduction">
```

[5]문서만들면서 알았는데 alt는 번역을 안시켰다.

```
<p class="photo"><img src="${img_asset_host}/images
    /pic_mc_photo.jpg" alt오상진="" /></p>
<div class="desc">
<dl class="overview">
<dt>Oh Sang Jin</dt>
<dd>The MC of Korea's first Dance Survival Program, 'Dancing9'!</dd>
</dl>
<ul class="filmography">
<li>Announcer </li>
<li>MBC's 'Complaint Zero', 'On Focus' MC</li>
<li>Appeared on MBC's 'Sunday Night', 'Dancing with the Stars' and
    many other variety shows</li>
</ul>
</div>
</div>
```

conv_translate.py

BeautifulSoup이라는 라이브러리로 *.kor.ftl안의 모든 노드를 깊이우선탐색하면서 text node일 경우만 csv의 한글값과 정확히 매칭하는 경우에 그 node를 교체한다.[6][7]

Listing 11: conv_translate.py

```
#!/bin/env python

import BeautifulSoup
import os, csv, codecs, cgi, re

def translate_html(srcfn, cb):
  fn = srcfn
  s = BeautifulSoup.BeautifulSoup(open(fn))
  dfs(s, cb)
  return s

def dfs(node, cb):
  if hasattr(node, 'contents'):
    for e in node.contents:
      dfs(e, cb)
  else:
    cb(node)

def read_translation_dic(f):
  r = unicode_csv_reader(f)

  translation_dic = {}
  for row in r:
    if len(row) == 5:
      kor,en,jp,chn,twn = row
      title = cgi.escape(kor.strip())
      title = re.compile('^\s*:\s*').sub('', title)
      translation_dic[title] = {
```

[6] dancing9 소스트리에만 있는데 원하는 분께는 직접 그 컴퓨터에 설치해드립니다.
[7] X로 만드는 것이 더 좋지 않을까요? 예 그것도 괜찮습니다.

```
            'eng': en,
            'jpn': jp,
            'chn': chn,
            'twn': twn
        }
    return translation_dic

def unicode_csv_reader(unicode_csv_data, dialect=csv.excel,
    **kwargs):
    # csv.py doesn't do Unicode; encode temporarily as UTF-8:
    csv_reader = csv.reader(utf_8_encoder(unicode_csv_data),
                            dialect=dialect, **kwargs)
    for row in csv_reader:
        # decode UTF-8 back to Unicode, cell by cell:
        yield [unicode(cell, 'utf-8') for cell in row]

def utf_8_encoder(unicode_csv_data):
    for line in unicode_csv_data:
        yield line.encode('utf-8')



def change_to_matching_str(dic, lang, node):
  s=node.string
  s=s.strip()
  if not s:
    return
  if dic.has_key(s):
    print dic[s][lang]
    #node.parent.string = dic[s][lang]
    node.replaceWith(dic[s][lang])
  else:
    print '---- not found:', s

def translate_ftl(dic, fmt):
  srclang = 'kor'
  srcfn = fmt % srclang
  for lang in ['eng','jpn','chn','twn']:
    targetfn = fmt % lang
    out = str(translate_html(srcfn, lambda s: change_to_matching_str
        (dic, lang, s)))
    print out
    open(targetfn, 'w').write(out)



def main():
  f = codecs.open('translation.csv', encoding='utf-8')
  dic = read_translation_dic(f)

  base = os.getcwd() + '/../../src/main/webapp/WEB-INF/ftl/'

  translate_ftl(dic, base + '/team/intro_con_body.%s.ftl');
  translate_ftl(dic, base + '/team/masterintro_con_body.%s.ftl');
  translate_ftl(dic, base + '/more/dancing_dic_con_body.%s.ftl');
  translate_ftl(dic, base + '/intro/mcintro_con_body.%s.ftl');
  translate_ftl(dic, base + '/vote/vote_message.%s.ftl');
```

```
    translate_ftl(dic, base + '/vote/solo_vote_message.%s.ftl');
85

  main()
```

정리하면

정리하면 ftl에서 번역이 필요할 때, if elseif로 쓸 수 있고, if 구문하나를 함수로 묶어 tranlation.ftl과 같은 파일에 모아두고 import해서 쓸 수도 있고, ftl이 너무 길어지고 중복되면 번역을 csv에 두고 csv에서 생성해서 쓸 수도 있다. ftl안에서 messageSource를 쓰고 싶으면 @spring.messageText를 사용한다. messageSource로 ResourceBundleMessageSource를 쓰면서 property파일에 번역을 저장해 쓸 수도 있고, me.interest.util.DatabaseMessageSource를 쓸 수도 있다. DatabaseMessageSource를 쓰려면 데이터베이스에 key와 번역값을 넣어야 하는데 INSERT 구문을 직접 타이핑해서 DB에서 실행할 수도 있고, csv로 모아두고 [8] 스크립트를 만들어 upsert [9] 할 수도 있고 admin을 두고(mwave와 인미는 그렇다) 사업팀에서 직접 입력하도록 할 수도 있다.

단어별 번역이 아닌 페이지 본문 수준의 번역이 필요할 때는 노드별 번역을 자동으로 하는 스크립트를 만들어 쓰는 것이 편했다.

언어, 국가코드

언어코드와 국가코드가 비슷할 때가 많기에 헷갈린다.

```
1   ko, kr, KOR, en, eng, US, GB, ja, JPN, zh, CHN, TWN, es, ESP.
```

우리가 보는 약자는 모두 ISO639이거나 ISO3166 이다.

ISO639는 전 세계의 언어 명칭에 고유 부호를 부여하는 국제 표준이다. ISO 639에서 2글자 코드와 3글자 코드를 둘 다 정의한다.

언어	2글자코드	3글자코드
한국어	ko	kor
영어	en	eng
일본어	ja	jpn
중국어	zh	? [10]
스페인어	es	spa [11]

ISO 3166은 전 세계의 나라와 부속 영토, 나라의 주요 구성 단위의 명칭에 고유 부호를 부여하는 국제 표준이다.

[8] DRM때문에 OpenOffice Calc를 깔아서 저장하였다.
[9] upsert - update를 먼저 해본 다음 아무row도 업데이트되지 않았으면 insert함.

국가	2글자코드	3글자코드
한국	KR	KOR
미국,영국	US,GB	?
일본	JP	JPN
중국,대만	CN,TW	CHN,TWN
스페인	ES	ESP

자바의 java.util.Locale 객체의 생성자에서는 2자리 코드만 입력받는다. mwave의 lang 변수에는 이름은 lang(언어)이나 ISO 3166(3글자 영토코드)중 하나로 KOR, ENG, JPN, CHN, TWN, ESP가 담겨있다. mwave의 langCountry 변수에는 IS0639와 ISO3166이 두 자리 코드로 엄격하게 담겨있다. [12]

중국어 - 간체(CHN), 번체(TWN) 구분

"오상진"을 간체로 "吴尚镇", 번체로 "吳尚鎮"으로 번역하였다. "오"자를 보면 간체가 좀 더 간단해보인다. 10자가 있으면 1자정도 간단해 보이는 글자가 섞여있다.

이 문서는

TEX문법으로 작성해 luaTeX으로 식자했습니다.

[12] mwave의 langCountry에 담긴 코드 == 인미에서 쓰는 코드 == UNIX의 LANG에 담긴 코드