

Özelleştirilebilir Java Tabanlı Betik Diller için Programlama Dili Alt Yapısı Geliştirme

Fırat KÜÇÜK¹, İbrahim ŞAHİN²

¹ Sakarya Üniversitesi, Adapazarı Meslek Yüksekokulu, Sakarya

² Düzce Üniversitesi, Teknik Eğitim Fakültesi, Elektronik ve Bilgisayar Eğitimi Bölümü, Düzce
fkucuk@sakarya.edu.tr, ibrahim.sahin@gmail.com

Özet: Programlama dilleri, bilişim sektörü başta olmak üzere tüm sektörlerde kullanım bulmakta ve bu sektörlerde ait uygulama geliştirme süresini kısaltan daha performanslı programlama dillerine her zaman ihtiyaç duyulmaktadır. Yaptığımız çalışmada uygulama geliştiricilere alışık olduğu dil imlasını kullanma yetisi sağlayan bir dil alt yapısı geliştirilmiştir. Geliştirilen dil alt yapısı, GPL lisansı altında serbestçe ve ücretsiz bir şekilde açık kaynak halde geliştiricilere ve kullanıcılara sunulmaktadır.

Anahtar Kelimeler: Programlama Dili Alt Yapısı, Söz dizim Çözümleyici, Özelleştirilebilir Programlama Dili

Developing A Programming Language Infrastructure For Customizable Java Based Scripting Languages

Abstract: Programming languages has an improving role in IT market. And programming languages that decrease application development duration is needed growingly. In our project, a programming infrastructure that provides the grammar reflects the application developer behaviors. Developed Language Infrastructure is represented to developers in GPL license for distributing it freely and free of charge.

Keywords: Programming Language Infrastructure, Language Parser, Customizable Programming Language

1. Giriş

Bilgisayar Dilleri kavramı, günümüzde kabına sığamamış cep telefonlarından kol saatlerine kadar tüm işlem birimi (Mikro işlemci, Mikro denetleyici, Sayısal İşaret İşleyici (DSP) gibi) bulunan elektronik cihazlarda kullanım alanı bulmuştur. Şüphesiz bu yönelim, programlanabilir bir elektronik aygıtın özelleştirilebilme büyüğünden ileri gelmektedir.

Programlanabilir bir aygıt, kullanıldığı sistem içindeki bütün girdi, çıktı ve etkileşim birimlerinin muhtemel tüm kullanım kombinasyonlarını kullanabilme yetisine sahiptir. Yüzlerce

mikroişlemci yönergesi ve onlarca tümleşik aygıtta sahip olan günümüz bilgisayarlarındaki bu kombinasyonlar, uygulama geliştiricilerin hayal gücü ile sınırlıdır.

Kasım 1954’de [1] ilk Yüksek Düzeyli Programlama Dili (High-Level Programming Language) FORTRAN’ın çıkması ile artık bilgisayarlar, daha kolay programlanabilir hale gelmiştir. Bu açılan yeni yüksek düzey programlama çağı ile birlikte artık programlar daha kısa bir süreç neticesinde yazılabilmektedir. Uygulama geliştiricilerin daha az kod ile daha çok iş yapma becerileri bu tarihten itibaren yeni çıkan paradigmalara paralel olarak katlanarak arttı.

FORTRAN'dan önce bir çıktı aygıtına bir kelime yazdırmak için kullanılacak yegâne yol; işlem birimini, makine dili yönergeleri ile ya da bu makine dili yönergelerine takma isimler verilerek oluşturulmuş Assembly dili sözcükleri ile beslemektir. Bu işlem, FORTRAN gibi bir üst düzey dilin icadı ile PRINT veya WRITE ifadesi kadar sadeleştirilebildi. FORTRAN makine diline derleme yaptığından sonuçtaki uygulama Assembly ile yazılan uygulamaya oldukça yakın başarımla (performans) sergilemekteydi. Artık programlar daha kısa sürede daha az kodlama ile yazılabiliyordu.

Bilgisayar dilleri, daha sonraları yalnız programlama kavramı içerisine sıkışıp kalmadı. Grafik Kullanıcı Arabirimi (GUI) tanımlama, veritabanı sorgulama gibi daha birçok alanda kullanım bularak uygulama geliştiricilerin daha çok işlem yapabilen programları, daha kısa sürede yazmalarına imkân tanıdı. Bilgisayar Dillerinin alt kolu olan Programlama dilleri ise artık yalnızca uygulama geliştirmek için değil uygulamaları özelleştirmek için de sıkça kullanılır hale geldiler.

Bunların yanında programlama dillerine ait sanal makineler (JVM, .NET Framework, vb...) ve farklı mikroişlemci platformları arası derlenebilen diller (Örn. C/C++) sayesinde, işlemci mimarilerine kolayca uyarlanabilen uygulamalar yazılmasına imkân tanınmıştır. Hiç şüphesiz bu yaklaşımlar da uygulama geliştirme sürecini olabildiğince kısaltmıştır.

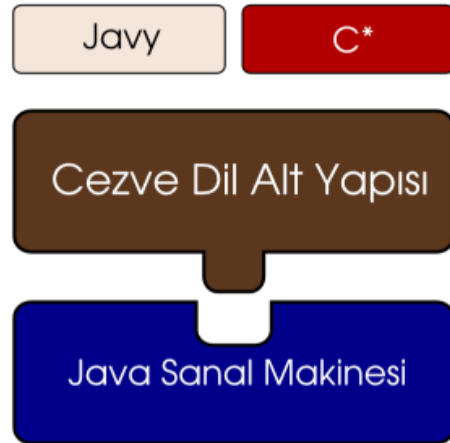
Sanal Makineler (Virtual Machines) sayesinde sanal makineye özgü yazılmış uygulamalar, sıfır hata ile desteklenen platformlar arası arasında taşınabilmektedir. Sanal makine üzerinde çalışmayan uygulamalar ise her platformda çalışan aynı programlama dilini derleyebilen derleyiciler sayesinde, farklı işlemci mimarileri üzerinde de çalışabilir hale gelmektedir. Örnek olarak; 86 milyon satır koda sahip Apple Mac OS X Tiger işletim sistemi, 210 gün gibi

bir süre içerisinde Power PC platformundan Intel platformuna taşınabildi. [2] Bu aktarımı gerçekleyen en büyük etmen hiç şüphesiz platformlar arası çalışabilen C derleyicileriydi.

Çalışmamız, Java Sanal Makinesi üzerinde çalışabilen yeni betik diller geliştirmeye olanak sağlayan ve bunları yorumlayan bir dil alt yapısını kapsamaktadır. Dil alt yapısı ile beraber yine açık kaynak çeşitli araçlar kümesi de sağlanmıştır. Sunulan çalışmanın farklı kollarında dil alt yapısı için tam kapsamlı Javy [3] adında konsept bir dil geliştirilmişti. Gerçekleştirilen bu çalışmada ise Javy ve benzeri dillerin kolayca geliştirilebileceği temel platform sunulmaktadır.

Bu platformdan ileride Cezve Dil Alt Yapısı veya kısaca Cezve diye bahsedilecektir. Cezve dil alt yapısı, platforma özel hazırlanmış ".g" uzantılı tek bir ANTLR [4] imlasını alıp tam teşekküllü bir betik dil oluşturabilmektedir.

2. Dil Alt Yapısı: Cezve



Şekil 1. Cezve dil alt yapısı mimarisi

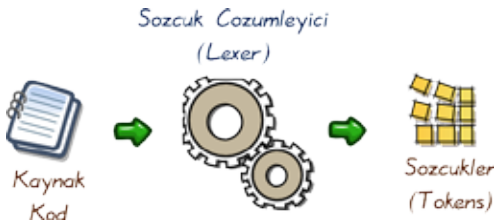
Cezve dil alt yapısı, Java Sanal makinesi üzerinde çalışmaktadır. Ve bu dil alt yapısı üzerinde koşan diller de alt yapı olarak Cezve'yi

kullanırlar. .NET CLI çatısının VB.NET, C# gibi dillere ev sahipliği yapması veya Groovy, Rhino, BSH, Jython gibi dillerin Java Sanal Makinesi üzerinde yer bulmaları gibi oluşturduğumuz konsept diller de ara katman olan Cezve üzerinde konumlanırlar.

Çalışmamızda oluşturduğumuz dil alt yapısının mevcut sürümünde dayandığı temel nokta ANTLR ayrıştırıcı oluşturucudur. Bunlara ila- veten Cezve, Java Sanal Makinesi ile iletişim kurabilecek birçok yönetici sınıf içermektedir. Yönetici sınıflar imla ile Java sanal makinesi arasında bir köprü niteliğindedir. Bir kaç KB miktarındaki bir BNF imla tanımı dosyasının karmaşık bir dil olma serüveni Cezve'nin kullandığı ANTLR kütüphanesinin bu imla dosyasından bir sözcük ayrıştırıcı oluşturması ile başlamaktadır.

2.1 Sözcük Çözümleme

Cezve, ANTLR Çözümleyici oluşturucu yardımıyla imla dosyasından Sözcük Çözümleyici (Lexer) oluşturur. Sözcük çözümleyici, BNF kurallarının sonlu belirgin otomat (DFA) haline dönüşmüş şeklidir. Çözümleyici oluşturulurken Cezve'nin *genelde* belirli bir düzenli ifadeye veya dizgeye karşılık gelmeyen anlamsal sözcükleri de çözümleyiciye eklenir. Bu anlamsal sözcükler, dil alt yapımız tarafından sunulan standart sözcüklerdir. Bu sözcüklerin (token) kullanılması halinde çözümleyicide çözümlenen bazı literaller daha sonraki aşamalarda Cezve tarafından kolayca anlaşılır halde olacaklardır.



Şekil 2. Sözcük çözümleme

Sözcük çözümleyici, bir dizge (string) kaynağından verileri anlamsal gruplar şeklinde ayırır. Bu ayırma neticesinde imla sözcükleri (tokens) oluşur. İmla sözcükleri Söz dizim ayrıştırıcı tarafından denetleneceklerdir.

Örnek bir Javy kodunun şu şekilde olduğunu düşünersek:

```
if (true) {  
    println("true");  
} else {  
    println("false");  
}
```

Sözcük çözümleyici bu karakter yığını şu sözcüklere ayırır:

```
'if'      : IF  
' '      : WHITESPACE  
'('      : LEFT_PARENTHESIS  
'true'   : CEZVE_LITERAL_BOOLEAN  
)'       : RIGHT_PARENTHESIS  
' '      : WHITESPACE  
'{'      : LEFT_CURLY_BRACKET  
'\n'     : EOL  
' '      : WHITESPACE  
'println' : CEZVE_ATOM_NAME  
'('      : LEFT_PARENTHESIS  
'"true"'  : CEZVE_LITERAL_STRING  
)'       : RIGHT_PARENTHESIS  
';'      : SEMICOLON  
'\n'     : EOL  
'}'      : RIGHT_CURLY_BRACKET  
' '      : WHITESPACE  
'else'   : ELSE  
' '      : WHITESPACE  
'{'      : LEFT_CURLY_BRACKET  
'\n'     : EOL  
' '      : WHITESPACE  
'println' : CEZVE_ATOM_NAME  
'('      : LEFT_PARENTHESIS  
'"false"' : CEZVE_LITERAL_STRING  
)'       : RIGHT_PARENTHESIS  
';'      : SEMICOLON  
'\n'     : EOL  
'}'      : RIGHT_CURLY_BRACKET  
'\n'     : EOL
```

Karakter yığını anlamlı imla sözcüklerine dönüştürülmüştür. Bu sözcüklerden CEZVE öneki ile başlayanlar ileriki safhalarda Cezve alt

yapısı tarafından anlamlı kabul edilecektir. Diğer sözcükler ise söz dizim çözümleyiciye gönderileceklerdir. Söz dizim çözümleyici de bunların bir kısmını kendi içerisinde yine Cezve standart sözcüklerine dönüştürebilir.

2.2 Söz Dizim Çözümleme

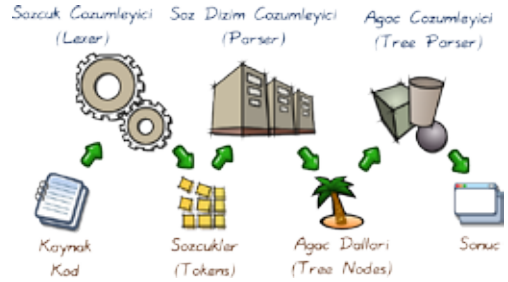
```
(CEZVE_SCRIPT
  (CEZVE_STATEMENT_IF
    (CEZVE_SUB_STATEMENT_IF
      (CEZVE_FRAGMENT_ATOM true)
      (CEZVE_STATEMENT_EXPRESSION
        (CEZVE_FRAGMENT_ATOM
          (CEZVE_FRAGMENT_METHOD
            println
              (CEZVE_FRAGMENT_METHOD_
                ARGUMENTS
                  (CEZVE_FRAGMENT_
                    EXPRESSION_LIST (CEZVE_FRAGMENT_ATOM
                      "true")))
                )
              )
            )
          CEZVE_SUB_STATEMENT_IF_END)
          (CEZVE_SUB_STATEMENT_ELSE
            (CEZVE_STATEMENT_EXPRESSION
              (CEZVE_FRAGMENT_ATOM
                (CEZVE_FRAGMENT_METHOD
                  println
                    (CEZVE_FRAGMENT_METHOD_
                      ARGUMENTS
                        (CEZVE_FRAGMENT_
                          EXPRESSION_LIST (CEZVE_FRAGMENT_ATOM
                            "false")))
                        )
                      )
                    )
                )
              )
            )
          )
          CEZVE_SUB_STATEMENT_ELSE_END)
          CEZVE_STATEMENT_IF_END)
  )
)
```

Cezve sistemine giren bir .g imla dosyasından ikinci olarak bir de söz dizim çözümleyici oluşturulur. Söz dizim çözümleyici de temelde bir Sonlu Belirgin Otomat DFA uyarlamasıdır. Sözcük çözümleyicinin çıktıları ile beslenen sözcük çözümleyici, sözcüklerin kurallı birer ifade olup olmadığını denetler. Söz dizim

çözümleyici, Cezve dil alt yapısının ağaç ayrıştırıcısı tarafından anlamlı olacak standart sözcükler üretir. Yukarıdaki örnekteki İmla sözcükleri söz dizim çözümleme sonrasında daha anlamlı sözcüklere dönüşür. Lisp tarzı bir ifade ile ağaç ayrıştırıcıya şu sıra düzensel yapı gönderilecektir.

Belirli bir ağaç yapısı şeklinde işlenen sözcükler, Cezve dil alt yapısının bütünleşik Ağaç Ayrıştırıcısına gönderilir.

2.3 Ağaç Çözümleme



Şekil 3. Ağaç çözümleme yapısı

Cezve dil alt yapısının standart sözcükleri halinde anlamsallaştırılan karakter katarı, ağaç çözümleyicinin yönetici sınıfları tarafından işlenirler. Ağacın en sonundaki atom veri tipleri Cezve dil alt yapısının sağladığı AtomManager tarafından tutulur. AtomManager bu ifadeleri ilişkili Java nesnelere dönüştüreceklerdir.

Örnekte adı geçen atom veri tiplerinden ikisi (CEZVE_FRAGMENT_ATOM "true") ve (CEZVE_FRAGMENT_ATOM "false") en baştan beri CEZVE_LITERAL_STRING tanımlaması ile ağaç ayrıştırıcıya ulaşır. Ağaç ayrıştırıcı da bu dizgeyi java.lang.String örneğine dönüştürür.

Cezve'nin dilin temel çatısını oluşturmak için kullandığı belirli başlı yönetici sınıflarını şöyle sıralayabiliriz.

- AssignmentManager: Atama İşlemleri
- AtomManager: Atom ve literallerin nesnelere dönüşüm işlemleri
- CastManager: Tip dönüşüm işlemleri
- ClassPathManager: Paket ve çalışma zamanı sınıf işlemleri
- ExpressionManager: İşlem grubu yönetim işlemleri
- OperatorManager: İşleç işlemleri
- StatementManager: Basit ve Bileşik ifade işlemleri
- TypeManager: Temel tip tanımlamaları

Cezve dil alt yapısının ağaç ayrıştırıcısı diğer ağaç adımlarını ilişkili yönetici sınıflara gönderir ve betiğin çalışma zamanında yorumlanmasını sağlar.

3. Sonuç

Cezve dil alt yapısı, uygulama geliştiricilere kendi alıştıkları imla ile betik programlar yazabilme imkânını sunmaktadır. Bu kapsamda uç birim, masaüstü ve hatta web tabanlı programlar yapılması da mümkün olmaktadır. Cezve, Web programları için işletim sisteminin standart giriş ve çıkış sistemi üzerinden CGI kipte çalışabilmesinin yanında çok kanallı (multithread) web uygulamalarını da desteklemektedir. Cezve dil alt yapısı, barındırdığı ön tanımlı Servlet'i kullanarak özelleştirilebilir imla ile yazılan uygulamaları Apache Tomcat gibi uygulama sunucularına taşıyabilmektedir.

Bunlara ilaveten Cezve, Java'nın temel ilkel tip örtücü sınıflarına müdahale etmektedir. java.lang paketinde tanımlanan belirli başlı ilkel tip örtücü sınıfları ve java.util paketindeki bazı koleksiyon tiplerini projemiz için uyarladık. Bu sayede oluşturulan diller Ördek Tip

Denetimi (Duck Typing), İşleç Aşırı Yükleme (Operator Overloading) gibi modern paradigmaları destekleyebilmektedir.

Dil alt yapısının sunduğu işleç sistemi, Python ile örtüşmektedir. Nesne bir işleç işlemine maruz kalıyorsa nesnenin işlece özel metotları çağrılmaktadır. Bu özel metotları da işleçler ile kullanabileceğimiz gibi mevcut nesnelere kalıtım yolu ile aldığımız alt metotları da kullanabiliriz. Cezve alt sistemi bulara ilaveten Python, PHP ve Perl gibi dillerin bazı işleçlerini de kendi bünyesine katmıştır. Bu manada yeni yapılan diller bu işleç özelliklerini de kendi bünyesine kolayca alabilirler.

Cezve dil alt yapısı ve beraberindeki konsept diller şu anda kararlı sürümüne ulaşmamış olmasına rağmen bir çok modern paradigmayı üzerinde barındırmaktadır. Cezve, Şu an CGR "Cezve Grammar" paketleri üzerinden kolayca dil transferi yapabilesinin yanında bütünlük kabuk ve dil yorumlayıcısı ile betik dil dosyalarını anında yorumlayabilme yetisine sahiptir.

4. Kaynaklar

- [1]. O'Reilly Media, Inc., www.oreilly.com/news/graphics/prog_lang_poster.pdf, History of Programming Languages
- [2]. Apple World Wide Developer Conference 2006, <http://www.apple.com/quicktime/qtv/wwdc06/>
- [3]. Javy Programming Language, <http://www.javy.org/>
- [4]. ANTLR - ANother Tool for Language Recognition, <http://www.antlr.org/>