# استفاده از مفاهیم برنامه نویسی صفت‌مبنا در محیط وب

سیدجمال‌الدین پیشوایی
pishvayi@ce.sharif.edu

مجیدرضا باقری
m_bagheri@ce.sharif.edu

علیرضا معیرزاده
moayerzadeh@ce.sharif.edu

دانشکده مهندسی کامپیوتر
دانشگاه صنعتی شریف

## چکیده

*برنامه نویسی صفت‌مبنا[1] که اولین بار در چارچوب کاری .NET. به صورت جدی مطرح شد با فراهم کردن امکان درج و مقداردهی صفات روی اِلمانهای زبان برنامه‌نویسی مانند کلاسها، متدها و فیلدها و بهره‌گیری از این صفات در زمان اجرا و یا زمان کامپایل موجب ساده‌تر شدن پیکربندی مؤلفه‌های نرم‌افزاری و سادگی استفاده از سرویس‌های زمان اجرا گشته است. همچنین این صفات می‌توانند به عنوان مبنای کار ابزارهای تولید کد استفاده شوند. اکنون نسخه‌های اخیر جاوا نیز از این امکان پشتیبانی می‌کنند. از همین ایده می‌توان در توسعه مؤلفه‌های وب که در آنجا بجای کلاسها، متدها و فیلدها با صفحات[2] و برچسب‌های داخل آنها سر و کار داریم نیز بهره جست و اقدام به درج، مقداردهی و نهایتاً بهره‌گیری از صفاتی که به این المان‌ها اختصاص می‌یابد نمود. در این مقاله به پیشنهاد راه‌کاری برای این مسأله خواهیم پرداخت. این ایده که تا کنون مطرح نشده است، می‌تواند موجب تحولی در زمینهٔ توسعه مؤلفه‌های وب گردد.*

***کلمات کلیدی:*** *برنامه‌نویسی صفت‌مبنا، توسعه مؤلفه‌های وب*

## مقدمه

برنامه‌نویسی صفت‌مبنا به برنامه‌نویسان امکان می‌دهد تا با اضافه کردن ابر داده[3] به اجزای مختلف یک برنامهٔ نرم‌افزاری مانند کلاس‌ها، متدها، فیلدها و مولفه‌ها رفتار آنها را بر مبنای این ابر داده‌ها تغییر دهند. به طور معمول ویژگی‌ها و قابلیتهای یک مؤلفه باید به صورت کامل در خودش پیاده‌سازی شود. اما گاهی محیطی که برنامهٔ ما در قالب آن اجرا می‌شود – مانند .Net Framework و یا Application Server های J2EE – به ما این امکان را می‌دهند تا از طریق ارائه توصیفاتی در مورد مؤلفه‌ها از بستر اجرایی بخواهیم تا برخی از این ویژگی‌ها و قابلیت‌ها را برای ما فراهم سازد. این توصیفات قالباً در فایل‌های پیکربندی درج می‌شوند. مثلاً در J2EE چنین توصیفاتی برای مؤلفه‌های EJB به صورت یک فایل XML به نام ejb-jar.xml که اصطلاحاً Deployment Descriptor

[1] Attribute Based Programming
[2] Page
[3] Metadata

خوانده می‌شود در اختیار Application Server قرار می‌گیرد. از طریق این فایل می‌توانیم برای کنترل دسترسی‌ها و مجازشناسی، استفاده بهینه از منابع، کنترل تراکنش‌ها و برخی موارد دیگر بدون نیاز به برنامه‌نویسی از Application Server سرویسهای را دریافت کنیم. برنامه‌نویسی صفت‌مبنا به ما این امکان را می‌دهد که بجای ویرایش فایلهای پیکربندی این توصیفات را به اِلمانهای زبان برنامه‌نویسی نسبت دهیم. مثلاً بجای درج ویژگی‌های تراکنشی متدهای یک EJB در Deployment Descriptor به هر یک از متدهای آن صفتی اختصاص دهیم که ویژگی تراکنشی آن متد را مشخص کند. به این ترتیب Application Server که در واقع بستر اجرای EJB محسوب می‌شود می‌تواند هنگام فراخوانی متدها – با استفاده از مکانیزم Reflection مقادیر صفات آنها را بازیابی کند و- با توجه به صفات آنها اقدام به مدیریت، باز کردن یا بستن تراکنشها نماید[4]. در صورت استفاده از برنامه‌نویسی صفت‌مبنا برای تغییر رفتار یک EJB نیاز به تغییر بیش از یک فایل ( کلاس مربوطه و فایل پیکربندی ) نخواهد بود و هرگونه تغییری در همان فایلِ کلاس اعمال می‌شود.

موارد کاربرد صفات، از آنچه که ذکر شد فراتر است. صفات درج شده روی مؤلفه‌های برنامه می‌توانند مبنای کار ابزارهای تولید کد[5] قرار گیرند که این یکی از عمده‌ترین موارد استفاده از صفات محسوب می‌شود. XDoclet که بی‌شک محبوب‌ترین و پراستفاده‌ترین ابزار تولید کد متن‌باز در جاوا محسوب می‌شود بر این مبنا عمل می‌کند.

می‌توان از ایده درج صفات روی اِلمانهای تشکیل دهندهٔ زبانهای برنامه‌نویسی در توسعه مؤلفه‌های وب که در آنها بجای زبانهای برنامه‌نویسی با زبانهای نشانه‌گذاری[6] و به خصوص HTML سر و کار داریم بهره جست. تاکنون به این ایده پرداخته نشده است اما به نظر می‌رسد در صورتی که امکانی فراهم شود که بتوانیم بر روی اجزای اصلی یک مؤلفه وب نیز شامل صفحات[7] و برچسب[8]های داخل هر صفحه صفاتی درج کنیم در مورد مؤلفه‌های وب نیز به دستاوردهای مشابهی خواهیم رسید.

در ادامه به معرفی برخی امکانات موجود برای برنامه‌نویسی صفت‌مبنا در زبانهای برنامه‌نویسی خواهیم پرداخت. سپس تلاش می‌کنیم با بیان چند کاربرد نمونه از دیدگاه برنامه‌نویسی صفت‌مبنا در توسعه مؤلفه‌های وب اهمیت و تأثیر وجود این امکان را روشن سازیم و نهایتاً به پیشنهاد راه‌کاری برای بوجود آوردن این امکان خواهیم پرداخت.

## معرفی برخی امکانات برنامه‌نویسی صفت‌مبنا

### .NET Framework

در .NET، امکان ایجاد و استفاده از صفت‌ها از نسخه اولیه آن یعنی نسخه ۱.۰ فراهم شده‌است. این بستر به ما اجازه می‌دهد تا صفت‌های مورد نظر خود را ایجاد کنیم و آن‌ها را به اجزای مختلف برنامه خود نسبت دهیم. کامپایلر

---

[4] برای اطلاع از مکانیزم دقیق استفاده از صفات بجای Deployment Descriptor به مرجع شماره یک مراجعه کنید.

[5] Code Generators

[6] Markup Language

[7] Page

[8] Tag

می‌تواند این صفات را در زمان کامپایل شناسایی کند و ویژگی‌های اعلام شده توسط صفات را اعمال نماید. برای نمونه به کد زیر توجه کنید:

```
using System.Runtime.CompilerServices;
public class Class1 {
        [MethodImpl(MethodImplOptions.Synchronized)]
        public void Method1() {
                // method implementation goes here
        }
}
```

در این مثال، صفت MethodImpl با پارامتر MethodImpl.Synchronized به متد با نام Method1 نسبت داده شده است. در .NET صفت MethodImpl برای تنظیم رفتار یک متد به کار می‌رود و پارامتر MethodImplOptions.Synchronized به کامپایلر اعلام می‌کند که بدنه این متد، یک ناحیه بحرانی[9] است. به این ترتیب کامپایلر خود کد مربوط به ناحیه بحرانی را ایجاد می‌کند و نیاز به دخالت برنامه‌نویس از بین می‌رود.

علاوه بر این، ایجاد صفات جدید در .NET بسیار ساده و همانند ایجاد دیگر کلاس‌ها است. تنها محدودیت این است که کلاس نشان‌دهنده صفت باید از کلاس System.Attribute مشتق شود. برای مثال صفت زیر را می‌توان برای مشخص کردن نویسندهٔ متدهای برنامه ایجاد کرد:

```
[AttributeUsage(AttributeTargets.Method)]
public class AuthorAttribute : System.Attribute {
        private string name;
        public string Name {
                get { return name; }
        }
        public void AuthorAttribute(string n){
                name = n;
        }
}
```

```
public class Class2 {
  [Author("Moayerzadeh")]
  public void Method2() {
    // ...
  }
}
```

مقادیر این صفات را می‌توان در زمان اجرا از طریق مکانیزم Reflection بازیابی نمود و از آن‌ها استفاده کرد. برای مثال با استفاده از صفت تعریف شده AuthorAttribute ، می‌توان برنامه‌ای ایجاد کرد که متدهای تشکیل دهنده یک کلاس را به همراه نام نویسندهٔ هر متد چاپ کند.

## تایپ‌های حاشیه‌نویسی[10] در J2SE 5.0

جاوا در نسخه ۵ خود امکان تعریف صفات و درج و مقداردهی آنها روی اجزای برنامه را به وجود آورد. این صفات در جاوا اصطلاحاً تایپ‌های حاشیه‌نویسی خوانده می‌شوند. J2SE 5.0 به ما امکان می‌دهد تایپ‌های حاشیه‌نویسی دلخواه خود را بنویسیم و در زمان اجرا – از طریق مکانیزم Reflection – یا در زمان کامپایل – مثلاً با کمک ابزاری به نام apt – این صفات را بازیابی کنیم و نسبت به آنها عکس‌العمل نشان دهیم. به این ترتیب می‌توان حجم کد مورد نیاز برای یک مؤلفهٔ نرم‌افزاری را تا حد قابل توجهی کاهش داد.

برای مثال، برای استفاده از سرویس JAX-RPC ، لازم است که واسط Remote توسط کلاس مربوطه پیاده‌سازی شود. بدون وجود صفت، کلاسهای زیر لازم است:

```
//PingIF.java

public interface PingIF extends Remote {
  public void ping() throws RemoteException;
}
```

```
//Ping.java

public class Ping implements PingIF {
        public void ping() {
                //...
        }
}
```

[9] Critical Section
[10] Annotation Types

اما در صورت وجودِ صفات، کار به مراتب ساده‌تر خواهد بود:

```
public class Ping {
        public @remote void ping() {
                //...
        }
}
```

برنامه با درجِ صفتِ @remote به کامپایلر[۱۱] – یا به عبارت دقیق‌تر Annotation Processor – اعلام می‌کنیم که پیاده‌سازی واسط Remote لازم است. به این ترتیب کامپایلر، پیاده‌سازی مناسب را فراهم می‌کند. تولید تایپهای حاشیه‌نویسی مانند remote نیز ساده‌است[۱۲]:

```
public @interface remote {
        //...
}
```

در ادامه خواهیم دید که ما نیز برای قرار دادن صفات در فایل‌های JSP دقیقاً از همین تایپ‌های حاشیه‌نویسی بهره خواهیم گرفت.

**XDoclet**

XDoclet – که یک ابزار تولید کد متن‌باز بسیار محبوب و پراستفاده محسوب می‌شود – زمانی به وجود آمد که امکان برنامه‌نویسی صفت‌مبنا در زبان جاوا وجود نداشت و توانست این امکان را تا حدی فراهم سازد. مبنای کار XDoclet به این شکل است که به برنامه‌نویس امکان می‌دهد صفات را به صورت doclet‌های javadoc که در واقع توضیح[۱۳] محسوب می‌شوند در بالای اجزای مختلف برنامه درج کند. برای پردازش صفات و تولید کد از روی آنها یکسری برچسب ANT[۱۴] وجود دارند که پس از اجرا با توجه به مقادیر صفات درج شده، اقدام به تولید کد می‌کنند. XDoclet ادعا می‌کند می‌تواند بیش از ۸۵ درصد کدهای مورد نیاز در J2EE از جمله فایلهای پیکربندی و Deploymnet Descriptor را تولید کند. برای اطلاعات بیشتر در این زمینه می‌توانید به مرجع شماره سه رجوع کنید.

از آنجا که ایدهٔ استفاده از صفات در وب ایده‌ای جدید است و صفات بوسیلهٔ تفسیرگرهای مؤلفه‌های وب قابل کامپایل نیستند برای استفاده از مفاهیم صفت‌مبنا در وب از راه‌حلی مشابه XDoclet بهره خواهیم گرفت.

## برخی کاربردهای برنامه‌نویسی صفت‌مبنا در وب

در این بخش به بیان اهمیت برنامه‌نویسی صفت‌مبنا در وب و ذکر برخی کاربردهای آن خواهیم پرداخت. ابتدا برخی از مهمترین کاربردهای برنامه‌نویسی صفت‌مبنا در زبانهای برنامه‌نویسی را بررسی می‌کنیم:

- **مشخص کردن محدودیت های امنیتی**: با قرار دادن صفاتی بر روی مؤلفه‌ها یا برخی متدهای آنها می‌توان مشخص کرد چه نقش های امنیتی مجاز به استفاده از یک مؤلفه یا یک متد می باشند.

---

۱۱ مقصود از کامپایلر مفهوم عام آن است نه برنامه‌ای که کدهای منبع جاوا را به بایت‌کد تبدیل می‌کند.

۱۲ برای اطلاع دقیق از نحوهٔ تعریف و استفاده از تایپ‌های حاشیه‌نویسی می‌توانید به مرجع شماره دو مراجعه فرمایید.

13 Comment

۱٤ ANT یک ابزار ساخت ( Build Tool ) چندسکویی مبتنی بر XML است که به زبان جاوا نوشته شده و عملکرد آن با make مشابهت دارد. برای اطلاعات بیشتر در این زمینه می‌توانید به سایتِ http://ant.apache.org مراجعه کنید.

- **مدیریت تراکنش‌ها**: به جای مدیریت تراکنش‌ها در متدهای مختلفِ یک مؤلفهٔ نرم‌افزاری، می‌توان با مشخص کردن ویژگی تراکنشی متدها در قالب صفات، مدیریت تراکنشها را به بستر اجرایی واگذار کرد.
- **قابلیت سریالی شدن**: برای ایجاد قابلیت سریالی شدن در کلاسها می توان به جای ایجاد مستقیم این قابلیت توسط برنامه‌نویسی در هر کلاس، وجود این قابلیت را به وسیله صفات به محیط اجرایی یا یک کتابخانه اعلام کنیم تا محیط اجرا یا کتابخانه مورد نظر نسبت سریالی کردن کلاس اقدام کند (این کار دقیقاً به همین صورت در چارچوب کاری .NET انجام می‌شود).
- **تولید کد**: همانطور که ذکر شد سادگی عملکرد ابزارهای تولید کد یکی از مهمترین آورده‌های برنامه‌نویسی صفت‌مبنا می‌باشد. در بسیاری از کاربردها، لازم است کدهایی بر اساس الگوهایی مشخص و از پیش تعیین شده که بر اساس کاربرد به صورت مختصر سفارشی می شوند ایجاد شوند. در این گونه موارد می توان به جای تولید این کدها به روش مستقیم، آنها را به کمک صفت معرفی نموده تا به صورت خودکار تولید شوند.
- **همگام سازی**[15]: ممکن است برخی متدهای یک کلاس ناحیه‌های بحرانی محسوب شوند. برای کنترل عدم همزمانی در فراخوانی این متدها در زبان جاوا از کلمهٔ کلیدی synchronized استفاده می‌شود. Virtual Machine کنترل می‌کند که همزمان دو ریسمان وارد ناحیه بحرانی نشوند. در .NET معرفی متدهایی که ناحیه بحرانی محسوب می‌شوند از طریق درج صفتی روی متد صورت می‌گیرد.

با این پیش زمینه می‌توانیم به بیان برخی از کاربردهای برنامه‌نویسی صفت‌مبنا در وب بپردازیم. مؤلفه‌های وب نیز معمولاً یک فایل پیکربندی دارند که با اعمال برخی تنظیمات در آن می‌توان سرویس‌هایی از بستر اجرایی آنها گرفت. مثلاً در J2EE ، تنظیمات هر مؤلفهٔ وب در قالب XML در فایلی به نام web.xml که Deployment Descriptor آن مؤلفه محسوب می‌شود، اعمال می‌گردد. از طریق این فایل می‌توان فایل‌های خوش‌آمدگویی[16] را مشخص نمود، تنظیمات امنیتی لازم برای هویت‌شناسی و مجازشناسی و محدود کردن دسترسی‌ها را اعمال نمود، فیلترها و Listenerها را معرفی کرد و برای انجام URL Mapping از آن بهره برد. ضمناً معمولاً برای توسعه مؤلفه‌های وب از امکاناتی مانند Struts[17] و Sitemesh[18] نیز بهره می‌گیریم که هر یک، پیکربندی خاص خود را دارند. به شکل مشابهی می‌توان به جای ویرایش فایلهای پیکربندی از درج صفات در صفحات وب بهره برد. موارد زیر را می‌توان به عنوان نمونه‌هایی از کاربردهای برنامه‌نویسی صفت‌مبنا در وب نام برد:

---

[15] Synchronization

[16] در صورتی که کاربر در مرورگر وب نام فایل خاصی را نیاورد – و به ذکر نام دایرکتوری اکتفا کند – خدمتگزار وب به طور پیش فرض یک فایل از دایرکتوری مربوطه ( مانند index.html یا index.jsp ) را به نمایش درخواهد آورد. به چنین فایلی اصطلاحاً فایل خوش‌آمدگویی (Welcome File ) گفته می‌شود.

۱۷ برای اطلاعات بیشتر به http://struts.apache.org مراجعه کنید.

۱۸ برای اطلاعات بیشتر به http://opensymphony.com/sitemesh مراجعه کنید.

- **مشخص کردن محدودیت‌های دسترسی**: برای مشخص کردن اینکه چه دسته از کاربران حق دسترسی به یک صفحه را دارند می‌توان صفاتی مانند PagePermission را برای صفحه مقداردهی نمود.
- **مشخص کردن فایلهای خوش‌آمدگویی**: به جای معرفی فایلهای خوش‌آمدگویی در Deployment Descriptor می‌توانیم از درج یک صفت به نام WelcomeFile روی صفحه استفاده کنیم.
- **URL Mapping**: چنانچه بخواهیم به ازای درخواست یک URL مانند calc/add/ فایلی مانند add.jsp نمایش داده شود باید از URL Mapping استفاده کنیم و آدرسِ calc/add/ را به add.jsp/ نگاشت کنیم. در حال حاضر این امکان با ویرایش Deployment Descriptor فراهم می‌شود. با استفاده از برنامه‌نویسی صفت‌مبنا می‌توان به کمک صفات این URL ها را مشخص نمود.
- **Exception Handling**: معمولاً در هر مؤلفه وب به ازای هر خطای HTTP به عنوان مثال پیدا نشدن یک صفحه (خطای ٤٠٤) ، یا هر نوع Exception ، صفحهٔ ویژه‌ای به کاربر نمایش داده می‌شود. برنامه‌نویس صفحات معادل خطاها و Exceptionها را در Deployment Descriptor مشخص می‌کند. به کمک برنامه‌نویسی صفت‌مبنا می‌توانیم با درج صفاتی روی این صفحات، مشخص کنیم هر یک به ازای چه خطایی باید نمایش داده‌شوند.
- **Caching**: در مؤلفه‌های وب می‌توان با استفاده از فیلتر برخی از منابع را cache کرد. برای سیاستگذاری این cache به ازای صفحات مختلف می‌توان از درج صفات بر روی صفحات موردنظر استفاده نمود.
- **Layout**: معمولاً هر گروه از صفحات یک مؤلفهٔ وب از یک ساختار (Layout) مشخص تبعیت می‌کنند و ظاهری مشابه دارند. ابزارهایی مانند Sitemesh با فراهم آوردن یک Template Service به ما کمک می‌کنند قسمت‌های مشترک صفحات را با هزینه کمتری تولید و مدیریت کنیم. برای مشخص شدن ساختار صفحات نیز می‌توان از درج صفات متناسب بر روی آنها استفاده نمود.

موارد کاربرد برنامه‌نویسی صفت‌مبنا به این موارد محدود نمی‌شود و برنامه‌نویس باید بتواند به دلخواه خود صفاتی تعریف و از آنها استفاده کند.

## تعریف و استفاده از صفات در وب

در زبان‌های برنامه‌نویسی شیءگرا اجزای اصلی زبان که روی آنها صفت درج می‌شود شامل متدها، فیلدها و خودِ کلاس‌ها می‌شود. در زبان‌های نشانه‌گذاری مانند HTML اجزای تشکیل دهنده که اختصاص صفت به آنها می‌تواند موضوعیت داشته باشد صفحات و برچسب‌های موجود در صفحات می‌باشند. باید راهکاری برای تعریف صفات، درج آنها روی صفحات و برچسب‌ها و پردازش آنها پیشنهاد گردد. در پیشنهاد ارائه شده، قواعد نحوی مورد استفاده بر مبنای فناوری Java Server Pages می‌باشد.

- **تعریف و مقداردهی صفت برای برچسبها**: هر برچسب در زبانهای نشانه‌گذاری می‌تواند دارای تعدادی صفت یا پارامتر باشد. به عنوان مثال برچسبِ مربوط به فونت در زبان HTML – به نام <Font> – دارای صفات رنگ و سایز است. بنابراین برای اختصاصِ صفات به یک برچسب می‌توان این صفات را در کنار صفات اصلی آن برچسب آورد. البته این روش دارای محدودیتهایی نیز هست

اما در هر صورت عمدهٔ نیازهای مورد نظر ما را مرتفع می‌سازد و از سادگی لازم نیز برخوردار است. در Tapestry[۱۹] نیز برای اضافه کردن مشخصهٔ شیءِ مرتبط با برچسب به شکل مشابهی صفتی به نام jwcid به آن برچسب اضافه می‌کنیم.

- **تعریف و مقداردهای صفت برای صفحه**: برای تعریف صفت‌هایی که می‌توان به صفحات وب اختصاص داد از همان تایپ‌های حاشیه‌نویسی در J2SE 5.0 استفاده می کنیم. درج صفات روی صفحه نیز در قالب توضیح[۲۰] صورت می‌گیرد ( مشابه XDoclet ). در مثال زیر یک تایپِ حاشیه‌نویسی به نام PagePermissions که می‌توان از آن برای محدود کردن دسترسی استفاده کرد تعریف شده و در یک صفحه از آن استفاده شده است:

```
//PagePermissions.java
public @interface PagePermissions{
        public String[] value();
}
```

```
//adminsOnly.jsp
<%---
   @PagePermissions({“admin”})
---%>
...
```

## نحوهٔ اعمال تنظیمات

برای اعمال تنظیمات مشخص شده توسط صفات باید پیش‌پردازشی بر روی صفحات انجام شود و سایر فایلهای مورد نیاز تولید شوند و یا برخی فایل‌ها ویرایش شوند. برای این کار مانند XDoclet از Taskهای ANT بهره می‌گیریم. این Taskها که هریک به برخی از صفات حساس هستند برمبنای مقادیر صفات اقدام به ایجاد یا ویرایش فایلها می‌کنند. نمونه‌ای از یک Build فایلِ ANT می‌تواند به صورت زیر باشد:

```
<project ... >
        <target ... >
                <web-compile>
                        <Welcome-Files include="**/*.jsp">
                        </Welcome-Files>
                </web-compile>
        </target>
</project>
```

این Task با مشاهده صفت WelcomeFile در یک صفحه، اقدام به ویرایش Deployment Descriptor و اضافه کردن آدرس آن صفحه در لیست فایل‌های خوش‌آمدگویی می‌نماید:

```
//index.jsp
<%---
   @welcomeFile
---%>
<html>
   ...
</html>
```

```
//web.xml
<web-app>
  <welcome-files>
    <welcome-file>/index.jsp</welcome-file>
 </welcome-files>
</web-app>
```

## نتیجه‌گیری

همان‌گونه که استفاده از دیدگاه برنامه‌نویسی صفت‌مبنا برای زبانهای برنامه‌نویسی تحول ایجاد کرد، می‌تواند برای زبانهای نشانه‌گذاری در وب نیز مؤثر باشد و برای برنامه‌نویسان سادگی، حجم کد کمتر و خوانایی و قابلیت

[۱۹] Tapestry یک چارچوب متن‌باز توسعه مؤلفه‌های وب می‌باشد. برای اطلاعات بیشتر به آدرسِ http://jakarta.apache.org./tapestry مراجعه نمایید.

نگهداری بیشتر را به ارمغان آورد. کاربردهای بسیاری برای برنامه‌نویسی صفت‌مبنا در وب وجود دارد. در این مقاله راه‌کاری برای استفاده از این دیدگاه در وب ارائه گردید. این راه‌کار مشابه راه‌کار XDoclet در مورد زبانهای برنامه‌نویسی و البته مبتنی بر تایپ‌های حاشیه‌نویسیِ J2SE 5 می‌باشد.

## کارهای آتی

در این مقاله به بیان کلی راه‌کار پرداخته شد. لازمست مشخصات دقیق[۲۱] این راه‌کار معین گردد و ابزارهایی برای پیاده‌سازی این امکان بوجود آیند. با توجه به ارتباط تنگاتنگ برنامه‌نویسی صفت‌مبنا و برنامه‌نویسی وجه‌گرا[۲۲] به نظر می‌رسد شبیه‌سازی امکانات وجه‌گرا نیز در وب می‌تواند موجب تحول شگرفی در حیطه توسعه برنامه‌های مبتنی بر وب ایجاد کند. ما در پروژه متن‌باز AspectWeb[۲۳] می‌کوشیم در آیندهٔ نزدیک این امکانات را فراهم کنیم.

## مراجع


[1] Java Community Process. 2004. Enterprise JavaBeans™ 3.0 Specification Draft (JSR 220). http://www.jcp.org/en/jsr/detail?id=220

[2] Java Community Process. 2004. A Metadata Facility for the JavaTM Programming Language Specification (JSR 175). http://www.jcp.org/en/jsr/detail?id=175

[3] The XDoclet Project home page. 2004. http://xdoclet.sourceforge.net


21 Specification
22 Aspect Oriented Programming
23 http://aspectweb.sourceforge.net