**به نام آنکه جان را فکرت آموخت**


**دانشگاه صنعتی شریف**
**دانشکده مهندسی کامپیوتر**
**مهندسی کامپیوتر - نرم‌افزار**

# بهبود بخش ساخت خودکار کد از خط تولید نرم‌افزار wise

**محل کارآموزی :**
**شرکت مشاوران نرم‌افزاری اعوان**

**گردآوردنده :**
**مجتبی ورمزیار**
**۸۷۱۰۹۳۳۴**

**استاد کارآموزی :**
**دکتر مرتضی امینی**

**تابستان ۱۳۹۲**

# فهرست عناوین

# ۱- محل و شیوه کارآموزی

شرکت اعوان در سال ۱۳۸۳ تاسیس شده است. این شرکت عضو شورای عالی انفورماتیک و سازمان نظام صنفی رایانه‌ای می‌باشد. راهبرد اعوان ارائه خدمات فوق تخصصی با تکیه بر گردآوری نخبگان علمی و فنی است.

محورهای اصلی فعالیت شرکت عبارتند از:

**تولید نرم‌افزارهای سفارشی**: رویکرد اعوان در این حوزه کاهش هزینه و زمان تولید با به کارگیری خط تولید نرم‌افزار اعوان (WISE Solution) و استفاده از تخصصی‌ترین فناوری‌ها همچون Java EE و Oracle است.

**خدمات آموزشی و مشاوره‌ای**: رویکرد اعوان در این حوزه استفاده از اساتید کارکاشته (Practitioner) و حمایت علمی از دانش‌آموختگان طی یک فرایند مدت‌دار آموزش و به کارگیری مهارتها است.

**تولید بسته های نرم‌افزاری**: رویکرد اعوان در این حوزه تولید محصولات تخصصی و راهگشا برای حل نیازمندیهای واقعی سازمانها و شرکتهاست.

## ۱.۱- اطلاعات تماس شرکت

آدرس : میدان توحید، خیابان توحید، پلاک ۳۷، طبقه سوم

شماره تماس : ۶۶۹۰۴۶۲۸ - ۶۶۹۰۴۸۱۷

شماره فکس : ۶۶۹۰۴۷۲۰

آدرس سایت : www.asta.ir

آدرس پست الکترونیکی : info@asta.ir

## ۱.۲- تعداد تقریبی کارکنان

تعداد کارکنان شرکت اعوان، که در محیط رسمی و اداری فعال می‌باشند حدودا ۱۵ نفر می‌باشد. البته شرکت تعدادی دیگر از نیروهای خود را برای فاز نگهداری[۱]، برای محصولات عرضه شده در شرکت های طرف قرارداد، به کار گرفته است. بنابراین می‌توان گفت محیط رسمی و اداری شرکت اعوان تقریباً کوچک محسوب می‌شود و این شرکت از بخش‌های گوناگون تشکیل نمی‌شود. البته این به منزله کوچک بودن حوزه های فعالیت شرکت نمی‌باشد و برای بررسی این معیار نیازمند بررسی محصولات شرکت و شرکت‌های طرف قرارداد هستیم.

۱ Maintenance

## ۱.۳- فهرست محصولات شرکت

محصولات این شرکت عبارتند از :

- مدیریت ارتباط با شهروند (ZRM)
- پورتال درون سازمانی ( WISE Portal)
- خط تولید نرم‌افزار (WISE Solutions)
- شبیه سازی شبکه اجتماعی
- سیستم مدیریت گزارش‌های پویا
- سیستم مدیریت اسناد
- ابزار نصب و به‌روز رسانی
- پورتال اینترنتی
- جدول همراه

# ۲- خلاصه کارهای انجام شده

شرکت اعوان برای تولید محصولات نرم‌افزاری خود، از یک خط تولید نرم‌افزار به نام WISE Solution استفاده می‌کند. خط تولید نرم‌افزار به تسریع فرایند ساخت و همچنین بالابردن کیفیت نرم‌افزار به عنوان یک محصول مهندسی کمک می‌کند. یکی از راهکارهای استفاده شده در این خط تولید برای تسریع فرایند ساخت نرم‌افزار، تولید خودکار کد از روی مدل ها است. تولید خودکار کد از روی مدل از جمله رویکردهای موجود در مهندسی نرم‌افزار برای ساخت سیستم‌های نرم‌افزاری می‌باشد که در حوزه معماری مبتنی بر مدل[۲] و ایجاد مبتنی بر مدل[۳] قرار دارد. نتایج قابل توجه این رویکرد از یک طرف کاهش هزینه و افزایش سرعت ساخت نرم‌افزار، و از طرف دیگر کاهش خطای انسانی را در ایجاد نرم‌افزار خواهد بود.

با توجه به وجود برخی ضعف‌ها و نبود برخی ویژگی‌ها در بخش تولید خودکار کد در خط تولید نرم‌افزار شرکت اعوان، مسئول فنی این شرکت (آقای پیشوایی) هدف کارآموزی من را بهبود و افزودن برخی ویژگی‌های جدید به این بخش از خط تولید نرم‌افزار شرکت اعوان قرار دادند.

بهبودها و ویژگی‌های افزوده شده به این خط تولید در دوره کارآموزی من عبارتند از :

۱. بهره بردن از maven archetype برای تولید پروژه قالب[۴] از پروژه های موجود در خط تولید و ساخت پروژه های جدید از روی پروژه های قالب از پیش ساخته شده.
۲. بهبود تولید خودکار مدل داده‌ای میانیبه زبان xml از روی نمودار کلاس[۵] (حاوی نوع موجودیت ها) به زبان UML.
۳. ساخت خودکار پروژه‌ها از پروژه های قالب از روی نمودار مؤلفه[۶] به زبان UML.
۴. بهبود مدل داده‌ای میانی برای ساخت خودکار کد کلاس‌های نوع موجودیت ها به زبان جاوا.

همچنین پیش از شروع به انجام این کارها، برای آشنایی بیشتر با فعالیت‌های فنی شرکت، با تکنولوژی های زیر آشنا شدم :

۱. Google Web Toolkit
۲. Spring

---

۲ Model Driven Architecture
۳ Model Driven Development
۴ Archetype project
۵ Class Diagram
۶ Component Diagram

۳. JavaEE

۴. Maven Plugin

در ادامه به کارهای انجام شده در هر هفته اشاره خواهم کرد. لازم به ذکر است کارهای ذکر شده در هر هفته، بیانگر کارهای انجام شده در روزهایی است که جمع ساعات گذرانده شده در شرکت در آن روزها، برابر با ۳۲ ساعت کاری بوده است و ممکن است آن روزها در یک هفته واقعی قرار نگرفته نباشند.

**هفته اول**

در این هفته به آشنایی با برخی از تکنولوژی های مورد استفاده در شرکت اعوان پرداختم. در حال حاضر، این شرکت از تکنولوژی Google Web Toolkit (یا به صورت خلاصه GWT) برای تولید پروژه های تحت وب خود بهره می جوید. بنابراین یادگیری این تکنولوژی ضروری بود.

در ادامه با تکنولوژی Spring آشنا شدم. این تکنولوژی پویایی بسیار بالایی به پروژه های java می‌دهد و امکان تغییر و تعریف دانه‌های جاوا[۷] را به سادگی و به کمک فایل‌های xml فراهم می‌آورد.

همچنین کمی به یادگیری JavaEE (که پیش از این نیز در درس برنامه سازی وب با آن آشنا شده بودم) پرداختم و و پس از نوشتن یک پروژه ساده، به کمک سرور Apache Tomcat، آن پروژه را در کامپیوتر شخص ام راه اندازی کردم.

**هفته دوم**

در این هفته نحوه کار با ابزار maven را یاد گرفتم. این ابزار دارای افزونه[۸] های گوناگون و فراوانی است که بسیاری از نیازهای پروژه‌های نرم‌افزاری را به صورت خودکار فراهم می‌آورد. همچنین امکان ساخت یک افزونه جدید برای این ابزاری برای کاربران وجود دارد. یکی از افزونه های این ابزار افزونه ای به نام create-from-project امکان ساخت پروژه قالب را از روی پروژه موجود فراهم می‌آورد . همچنین افزونه ای دیگر به نام generate وجود دارد که امکان ساخت یک پروژه را از یک پروژه قالب (که از پیش در مخزن پروژه های قالب وجود دارد) فراهم می‌آورد (یعنی عکس عمل پیشین). همچنین پس از آشنایی با maven به افزودن ویژگی ساخت پروژه قالب و ساخت پروژه جدید از روی یک پروژه قالب پرداختم. از آنجایی که نیاز بود کمی با خط تولید نرم‌افزار WISE Solution و ساختار آن آشنا شوم تا بتوانم ویژگی مورد نظر را به آن بیافزایم، پس از این آشنایی با ساختار خط تولید، ویژگی مورد به کمک maven به خط تولید نرم‌افزار اضافه کردم.

**هفته سوم**

در این هفته می‌بایستی کار انجام شده در هفته پیشین را به یکی از افراد شرکت تحویل می‌دادم. پس از تحویل اولیه کار، برخی ایرادات موجود در ویژگی افزوده شده توسط مسئول شرکت گرفته شد. بنابراین بخشی از زمان این هفته به اصلاح موارد پیاده‌سازی شده در هفته گذشته صرف شد.

---

۷ Java beans

۸ Plugin

کار دیگری که در این هفته آغاز شد، بهبود ویژگی خواندن نمودار کلاس و ساخت خودکار مدل داده‌ای میانی در خط تولید نرم‌افزار بود. برای این کار می بایستی ابتدا با نحوه خواندن و استخراج عناصر نمودار کلاس به کمک کتابخانه emf.uml آشنا می‌شدم. علاوه بر یادگیری با نحوه خواندن عناصر یک نمودار کلاس به زبان UML (شامل کلاس، ویژگی[۹]ها، و روابط[۱۰])، با نحوه خواندن کلیشه[۱۱] ها در نمودار کلاس نیز می‌بایستی آشنا می‌شدم . زیرا برای ساخت مدل میانی، نیاز به افزودن برخی اطلاعات افزونه بودیم که این اطلاعات را می توانستیم از طریق مقادیر برچسب‌ها[۱۲] (که آن نیز از طریق کلیشه ها ممکن است) به نمودار کلاس بیافزاییم.

**هفته چهارم**

در این هفته به ادامه بهبود ویژگی دوم (که در هفته پیش آغاز شده بود) پرداختم. بنابراین نخست خواندن نمودار کلاس و عناصرش را پی گرفتم. همچنین، برای اطلاعات افزونه مدل داده‌ای میانی، می‌بایستی کلیشه ها و برچسب‌های مقداردارشان را برای دربرگیری اطلاعات افزونه در نمودار کلاس، طراحی می‌کردم. در نتیجه بخشی از این هفته نیز به این موضوع پرداخته شد. لازم به ذکر است در این شرکت از نرم‌افزار Visual Paradigm برای رسم نمودارهای UML استفاده می‌شود و می بایستی کلیشه های طراحی شده به گونه‌ای، که قابل وارد کردن به این نرم‌افزار باشند، در اختیار مدلسازان قرار گیرد. برای این کار از امکانات خود این نرم‌افزار برای استخراج کلیشه ها در قالب فایل xml استفاده شد.

پس از یافتن تسلط در خواندن نمودار کلاس و طراحی کلیشه ها، شروع به طراحی کلاس‌ها برای ایجاد یک ساختار منظم که دربرگیرنده اطلاعات موجود در نمودار کلاس بود کردم. این کار برای منتزع کردن ساخت مدل میانی از خواندن ویژگی‌های نمودار کلاس UML بود. با بهره گیری از این ساختار به سادگی ویژگی‌های مورد نظر از نمودار کلاس استخراج می‌شد و در اختیار سازنده مدل داده‌ای میانی (که اصلی‌ترین بخش آن یک فایل groovy بود) قرار می‌گرفت.

**هفته پنجم**

در این هفته به ادامه طراحی کلاس‌هایی پرداختم که وظیفه نگهداری تمامی اطلاعات نمودار کلاس را (عناصر نمودار به همراه برچسب‌های مقدار دار) برعهده داشتند. پس از طراحی این کلاس‌ها به پیاده‌سازی آن‌ها پرداختم. همچنین در طی طراحی مجبور شدم کمی طراحی ام را تغییر و بهبود دهم. این تغییر طراحی برای استفاده راحت‌تر و بهتر برای ایجاد خودکار مدل میانی بود. در این هفته ویژگی مورد نظر در خط تولید نرم‌افزار بهبود یافت و با استفاده از یک نمودار کلاس (که شامل کلاس‌های داده‌ای می‌باشند و معمولاً توسط مدلساز پروژه ایجاد می‌شوند) امکان ساخت خودکار مدل میانی فراهم آورده می‌شود. با توجه به اینکه خط تولید نرم‌افزار از پیش دارای ویژگی تولید خودکار کد کلاس‌ها به زبان جاوا از روی مدل میانی بوده است، با بهبود این ویژگی، عملاً یک امکان مناسب برای مدلساز فراهم می‌شود و آن ساخت نرم‌افزار از روی یک نمودار کلاس است. این امکان فرایند ساخت نرم‌افزار را هم تسریع می‌بخشد و

۹ Attribute
۱۰ Relationship
۱۱ Stereotype
۱۲ Tagged value

همچنین موجب کاهش خطای برنامه سازی می‌شود.

علاوه بر این موضوع در این هفته به اصلاح برخی موارد موجود در کار نخستین (ویژگی پیاده‌سازی شده با maven) پرداختم و ویژگی مورد نظر با همکاری یکی از کارکنان شرکت، به طور کامل به خط تولید نرم‌افزار افزوده شد.

**هفته ششم**

در این هفته شروع به افزودن یک ویژگی جدید در خط تولید نرم‌افزار کردم. این ویژگی براساس ویژگی نخستین (یعنی ساخت یک پروژه جدید از روی پروژه های قالب) تعریف شده بود. آقای پیشوایی قصد داشتند با توجه به وجود امکان ساخت خودکار پروژه از پروژه های قالب، این کار از روی یک نمودار مؤلفه صورت پذیرد. مشابه خواندن نمودار کلاس به زبان UML، در اینجا قرار شد که یک نمودار مؤلفه خوانده شود و با بهره گیری از برچسب های یک کلیشه برای هر مؤلفه، پروژه ها به صورت خودکار ساخته شوند. در اینجا هر مؤلفه نمایانگر یک پروژه است و مقدار بچسب ها ویژگی‌های پروژه ای را، که قرار است از روی یک پروژه قالب ساخته شود، مشخص می نمایند.

برای این کار علاوه بر طراحی کلیشه مورد نظر (به همراه برچسب هایش)، نیاز بود یک افزونه برای ابزار maven نوشته شود که کار کلی خواندن نمودار و ساخت پروژه را برعهده گیرد. در اینجا می بایستی با نحوه فراخوانی یک افزونه درون یک افزونه دیگر آشنا می‌شدم (زیرا ساخت پروژه از پروژه قالب با افزونه generate از ابزار maven امکان‌پذیر است).

در این هفته این ویژگی پیاده‌سازی شد و در خط تولید نرم‌افزار قرار داده شد.

**هفته هفتم**

در این هفته کار جدیدی را شروع کردم. این کار در ادامه کار دوم بود و هدفش بهبود بخش ساخت خودکار کد از روی مدل داده‌ای میانی بود. بخشی از خط تولید نرم‌افزار،وظیفه ساخت خودکار کد را از روی مدل داده‌ای میانی برعهده دارد؛ اما ظاهراً این بخش از نرم‌افزار به خوبی پیاده‌سازی نشده و نیاز به مهندسی مجدد دارد. مدل میانی در اینجا یک فایل xml است که بسیاری از عناصر و برچسب[۱۳]های آن به طور مستقیم و در بخش‌های مختلف ساخت خودکار کد مورد استفاده قرار گرفته است. این نحوه استفاده و خواندن فایل xml معایبی از جمله انعطاف پذیری بسیار پایین (در صورت تغییر در مدل میانی) و همچنین پیچیدگی بسیار بالا را به همراه داشته است. آقای پیشوایی پیشنهاد دادند که ابتدا یک متامدل از روی مدل میانی استخراج شود و کلاس‌های این متامدل پیاده‌سازی شوند. سپس در بخش ساخت خودکار کد از روی مدل میانی، نخست مدل میانی خوانده شود و در کلاس‌های متامدل طراحی شده ریخته شود و سپس بخش ساخت خودکار کد از این متامدل و کلاس‌هایش برای ساخت خودکار کد بهره بجوید.

در این هفته تلاش شد ابتدا یک متامدل از مدل میانی موجود استخراج شود و به کمک ECore پیاده‌سازی شود. ECore امکان پیاده‌سازی متامدل را فراهم می‌آورد، به گونه‌ای که امکان ساخت خودکار کلاس‌های متامدل را نیز فراهم می‌آورد. درنتیجه هزینه اصلی طراحی و پیاده‌سازی آن به کمک ECore خواهد بود.

---

۱۳ tag

**هفته هشتم**

در این هفته به ادامه کار هفته‌ی هفتم پرداختم.

# ۳- ارزیابی کارآموزی

کارآموزی در شرکت اعوان بسیار برای من آموزنده بود. اصلی‌ترین عامل این امر را نیز در نحوه نگرش شرکت به فرایند ساخت نرم‌افزار می‌دانم. اعوان برخلاف بسیاری از شرکت ها، با بهره گیری از خط تولید نرم‌افزار، توانسته فرایند ساخت نرم‌افزار را بهبود بخشد که از نتایج آن می‌توان به کاهش هزینه‌های ساخت نرم‌افزار و افزایش کیفیت آن اشاره کرد. قرار گرفتن در محیطی که به واقع نرم‌افزار را یک محصول مهندسی می‌انگارد و آن را نیز به طور مهندسی می‌سازد، برای من بسیار آموزنده بود.

افزون بر این، موضوع کارآموزی من نیز در جهت بهبود خط تولید نرم‌افزار بود که این خط تولید یکی از اصلی‌ترین عوامل در بهبود ساخت نرم‌افزارها است. این موضوع از علاقه های شخصی من نیز بوده است؛ البته رویکردی مهم در مهندسی نرم‌افزار نیز به شمار می‌رود، زیرا مهندسان نرم‌افزار درصددند که با ساخت خودکار نرم‌افزار از روی مدل‌ها، هزینه و خطای ایجاد نرم‌افزار را کاهش و همچنین سرعت ساخت نرم‌افزار را افزایش دهند.

همچنین آشنایی با برخی تکنولوژی‌ها از جمله maven ، که امکانات بسیاری را برای تسریع ساخت نرم‌افزار در اختیار کاربر قرار می‌دهد، برای من سودمند بود. Maven ابزاری بسیار مفید است که استفاده از آن (یا ابزارهای مشابه که من از وجود آن‌ها ناآگاهم) در پروژه ها بسیار ضروری به نظر می‌رسد، به گونه‌ای که در محصولات پیچیده، از بهره نبردن از چنین ابزارهایی گریزی نیست. همچنین آشنایی کوتاه با تکنولوژی GWT و Spring نیز برای من سودمند بوده است.

محیط آرام و مناسب کاری از دیگر موارد قابل توجه شرکت اعوان است. محیط آرام و البته دوستانه شرکت موجب خرسندی من بود. همچنین همکاری همکاران شرکت (که عمدتا با تجربه می باشند) در کمک به من، برای تکمیل دانسته‌ها در حین کارآموزی، برایم رضایت‌بخش بود. من شرکت اعوان را مکانی بسیار مناسب برای کارآموزان مهندسی نرم‌افزار می‌دانم و به دیگر کارآموزان، این شرکت را به عنوان شرکتی بسیار مناسب و با کیفیت پیشنهاد می‌کنم.

# ۴- گزارش فنی

در این بخش به مواردی خواهم پرداخت که در دوره کارآموزی آن‌ها را فراگرفته‌ام. این موارد شامل برخی نکات فنی کارهای انجام شده نیز است. در هر زیربخش از این بخش، یکی از کارهای انجام شده را شرح خواهم داد.

## ۴.۱- کار با GWT

GWT ابزاری برای ساخت پروژه های تحت وب است. این ابزار امکانات بسیاری را برای کاربر فراهم می‌کند تا به سادگی بتواند یک پروژه تحت وب ایجاد کند. از امکانات ویژه ای که این ابزار در اختیار قرار می‌دهد، جدا کردن برنامه ساز از پیچیدگی های کدهای javascript ،HTML، و css می‌باشد. در این ابزار کاربر به زبان جاوا برنامه‌سازی می‌کند و در فرایند کامپایل پروژه، کدهای javascript ،HTML، و css به صورت خودکار، توسط ابزار، از روی کدهای جاوا ساخته می‌شود.

برای طراحی عناصر صفحات وب، می‌توان متغیرهایی از کلاس‌های کتابخانه این ابزار تعریف نمود. در کتابخانه‌ی این ابزار، متناظر با هر عنصر در صفحات وب به زبان HTML (مثل برچسب‌های label , table, button و ...)، کی کلاس وجود دارد. مثلاً کلاس Button، برای دکمه استفاده می‌شود. اما نکته قابل توجه در این ابزار، استفاده بسیار ساده از این کلاس‌هاست، که علت این امر برنامه‌سازی به زبان سطح بالاتر جاوا نسبت به زبان HTML است. همچنین نرم‌افزار eclipse دارای افزونه‌هایی برای کار با این ابزار است که امکان اشکال‌زدایی[۱۴] از کدهای جاوای پروژه را به همراه می‌آورد.

پس از نصب افزونه‌ی این ابزار در محیط eclipse، می‌توان یک پروژه جدید از نوع GWT ساخت. پروژه ایجاد شده، شامل کلاس‌هایی آماده است که به صورت خودکار ساخته شده‌اند که برای طراحی صفحات و ایجاد ارتباط بین خدمتگزار[۱۵] و کاربر[۱۶] می‌توان از آن‌ها استفاده کرد. پروژه ساخته شده توسط این افزونه شامل سه بسته[۱۷] اصلی زیر می‌باشد :

۱. client
۲. server
۳. shared

در ادامه، نخست به نحوه طراحی صفحات وب خواهیم پرداخت. سپس به نحوه ارتباط بین خدمتگزار و کاربر خواهیم

۱۴ Debugging
۱۵ Server
۱۶ Client
۱۷ Package

پرداخت.

### ۴.۱.۱- طراحی صفحات وب

در بسته client یک کلاس همنام با نام پروژه ساخته شده وجود دارد (برای مثال اگر نام پروژه testGWT گذاشته شده باشد، نام این کلاس TestGWT خواهد بود). برای طراحی صفحه وب مورد نظر، کافی است تابع onModuleLoad از این کلاس را تغییر داد. این تابع در حین بارگذاری صفحه فراخوانی می‌شود. با تعریف متغیر از کلاس‌های کتابخانه GWT در این کلاس، که متناظر با عناصر صفحات وب می‌باشد، و مقدار دهی آن‌ها در این تابع، صفحه مورد نظر را می‌توان طراحی کرد.

در اینجا کتابخانه‌ی این ابزار یک کلاس به نام RootPanel وجود دارد که در حکم برچسب body در صفحه به زبان HTML می‌باشد. این کلاس دارای یک تابع ایستا[۱۸] به نام get است. با فراخوانی این تابع، یک شیء که در حکم بدنه اصلی صفحه است، برگردانده می‌شود. سپس با تعریف متغیرها و فراخونی تابع add بر روی این شیء، عناصر تعریف شده را می‌توان در صفحه وب قرار داد. برای مثال اگر بخواهیم یک دکمه را در صفحه قرار دهیم، می‌توانیم به صورت زیر عمل کنیم :

```
Button b = new Button();
b.setText (“Test Button”);
RootPanel.get().add(b);
```

به این ترتیب یک دکمه با محتوای متنی Test Button در صفحه قرار می‌گیرد.

همچنین، کتابخانه‌ی این ابزار دارای دو کلاس مهم به نام VerticalPanel و HorizontalPanel است. به کمک این دو کلاس می‌توان عناصر را در صفحه اصلی وی بچینیم. عناصری که به یک شیء از کلاس VerticalPanel افزوده می‌شود، به صورت عمودی در کنار هم قرار می‌گیرند، و عناصر اضافه شده به شیء از کلاس HorizontalPanel، به صورت افقی در کنار هم قرار می‌گیرند.

### ۴.۱.۲- ارتباط با خدمتگزار

برای ایجاد ارتباط بین کاربر و خدمتگزار، می‌بایستی ابتدا یک واسط در بسته client تعریف نمود. این واسط می‌بایست واسط RemoteService را گسترش دهد[۱۹]. همچنین این واسط دارای سرایند یک تابع می‌باشد. این تابع درواقع همان تابعی است که از طرف کاربر فراخوانی می‌شود و پس از اجرا در سمت خدمتگزار، خروجی آن به کاربر فرستاده می‌شود. یک نمونه واسط می‌تواند به صورت زیر باشد :

---

۱۸ static

۱۹ extends

```
@RemoteServiceRelativePath("greet")
public interface GreetingService extends RemoteService {
        String greetServer(String name) throws IllegalArgumentException;
}
```

این واسط دربرگیرنده یک تابع می‌باشد که در سمت کاربر با یک رشته به عنوان پارامتر ورودی فراخوانی می‌شود و به سمت خدمتگزار فرستاده می‌شود. پس از اجرا و پردازش تابع با پارامتر ورودی در سمت خدمتگزار، یک رشته به صورت خروجی به سمت کاربر فرستاده می‌شود.

پس از تعریف این این واسط، نیازمند تعریف یک کلاس در بسته server هستم، که این کلاس می‌بایستی واسط تعریف شده را پیاده‌سازی کند. از آنجایی که این تابع در سمت خدمتگزار اجرا می‌شود، می‌تواند شامل هر منطقی، از جمله اتصال به پایگاه داده و بررسی یک قانون، باشد.

پس از تعریف واسط و کلاس پیاده‌ساز آن، یک واسط دیگر نیز در بسته client تعریف می‌شود. این واسط مشابه واسط تعریف شده با همان توابع می‌باشد، با این تفاوت که خروجی توابع همواره void است و یک پارامتر، افزون بر پارامترهای اصلی توابع، در هر تابع وجود دارد. این پارامتر یک شیء از کلاس کلی[۲۰] AsyncCallback است، که نوع کلی[۲۱] آن، همان خروجی تابع است. برای مثال :

```
public interface GreetingServiceAsync {
        void greetServer(String input, AsyncCallback<String> callback)
                        throws IllegalArgumentException;
}
```

با توجه به اینکه خروجی تابع greetServer در واسط اولیه، کلاس String می‌باشد، نوع کلی در آخرین پارامتر از جنس کلاس String می‌باشد (شیء به نام callback).

حال برای فراخوانی این تابع در سمت کاربر (برای مثال به واسطله فشردن یک دکمه در صفحه وب)، کافی است ابتدا یک شیء از واسط دوم به صورت زیر تعریف شود :

```
private final GreetingServiceAsync greetingService = GWT
                        .create(GreetingService.class);
```

حال با داشتن این شیء، می‌توان تابع greetServer را با پارامتر ورودی فراخوانی کرد. برای فراخوانی این تابع، علاوه بر پارامترهای اصلی که با توسط کاربر مشخص خواهد شد (برای مثال با توجه به مقادیر ورودی‌های یک فرم در صفحه وب)، می‌بایستی پارامتر نهایی از جنس کلاس AsyncCallback را مقداردهی کنیم. این کلاس دارای دو تابع با نام های onFailur و onSuccess است. این دو تابع، با توجه به پاسخی که از سمت خدمتگزار به سمت کاربر ارسال می‌شود فراخوانی می‌شوند. در صورتی که پاسخ خدمتگزار موفقیت آمیز باشد، تابع onSuccess فراخوانی خواهد شد. در

۲۰ Generic

۲۱ Generic Type

غیر این صورت تابع onFailur، به منزله رخداد یک خطا، فراخوانی خواهد شد. با گرفتن یک شیء از این کلاس و پیاده‌سازی این دو تابع برای آن، می‌توان پاسخ گرفته شده از سمت خدمتگزار را در سمت کاربر، در صفحه وب به کاربر نمایش داد.

## ۴.۲- کار با Maven

Maven ابزاری است که امکانات فراوانی را برای پروژه‌های نرم افزاری در اختیار کاربران قرار می‌دهد. این امکانات گوناگون از طریق افزونه‌های گوناگونی است که برای این ابزار وجود دارد افزونه‌های مهمی که من در این پروژه از آنها استفاده کردم عبارتند از create-from-project و generate. در زیر کارکرد و نحوه استفاده از هر کدام از این افزونه‌ها را اشاره خواهیم کرد.

### ۴.۲.۱- افزونه create-from-project

این افزونه توانایی ساخت یک پروژه قالب از روی یک پروژه موجود را دارد. لازم به ذکر است که پروژه مورد نظر می‌بایستی از نوع پروژه‌های maven ای باشد. این پروژه‌ها دارای یک ساختار استانده[۲۲] و یک فایل به نام pom.xml هستند. مشخصات اصلی پروژه‌های maven، از جمله groupId و artifactId، در این فایل مشخص می‌شود.

پروژه قالب، پروژه ای است که از متغیر کردن برخی ویژگی‌های یک پروژه موجود، به منظور استفاده مجدد از آن پروژه، بدست می‌آید. از جمله این ویژگی‌ها می‌توان به بسته اصلی[۲۳]، artifactId ،groupId، و حتی نام کلاس‌ها و توابع، اشاره نمود.

برای کار با این افزونه به صورت زیر می‌توان عمل کرد :

```
mvn archetype:create-from-project
```

در صورتی که در برنامه Terminal در سیستم عامل لینوکس، در مسیر[۲۴] یک پروژه از نوع maven قرار داشته باشیم، با زدن این دستور، یک پروژه قالب از آن پروژه ساخته می‌شود.

پس از ساخت پروژه قالب از این پروژه، می‌توان با دستور زیر، پروژه قالب را برای استفاده مجدد، در مخزن پروژه های قالب این ابزار نصب کرد :

```
mvn install
```

این دستور می بایستی در مسیری که پروژه قالب ساخته شده است اجرا گردد.

---

۲۲ Standard

۲۳ Root package

۲۴ directory

### ۴.۲.۲- افزونه generate

این افزونه برای ساخت یک پروژه از روی یک پروژه قالب موجود در مخزن است. با دادن ویژگی‌های پروژه قالب (شامل groupId، artifactId، و نسخه پروژه قالب) و ویژگی‌های پروژه جدید، یک پروژه از روی پروژه قالب ساخته می‌شود. این دستور به صورت زیر می‌باشد :

```
mvn archetype:generate -DarchtypeGroupId=... -DarchetypeArtifactId=... -DarchetypeVersion=... -DprojectGroupId=... -DprojectArtifactId=... -DprojectVersion=...
```

همانطور که مشاهده می‌شود، این دستور شامل پارامترهایی که است پروژه قالب را مشخص می‌کند. همچنین برخی پارامترهای این دستور برای مشخص نمودن ویژگی‌های متغیر پروژه قالب در پروژه جدید است.

### ۴.۲.۳- نحوه پیاده‌سازی در خط تولید نرم‌افزار

برای پیاده‌سازی این دو امکان در خط تولید نرم‌افزار، از ابزار Ant استفاده شد. Ant امکان اجرای دستورات maven و افزونه‌هاش را فراهم می‌آورد. برای اجرای یک دستور maven در فایل build.xml (که مربوط به ابزار Ant است)، به صورت زیر عمل می‌کنیم :

```
<artifact:mvn pom=”pom address”...>
        <arg value=”archtype:create-from-project />
        <arg value=”... />
        …
</artifact:mvn>
```

همانطور که مشاهده می‌شود، با برچسب arg، می‌توان نوع افزونه و دیگر پارامترهای دستور را مشخص نمود.

## ۴.۳- ساخت خودکار مدل داده‌ای میانی از روی نمودار کلاس

ساخت مدل داده‌ای میانی (در قالب فایل xml) از روی نمودار کلاس و با توجه به کلیشه‌ها، از دیگر مواردی بود که در دوره کارآموزی بنده صورت گرفت.

نخست به طراحی کلیشه ها و برچسب‌های مقداردار (به منظور قرار دادن اطلاعات افزونه در نمودار کلاس) پرداختم و پس از آن به تعریف آن‌ها با استفاده از نرم‌افزار visual paradigm پرداختم. در مدل داده‌ای میانی ۴ عنصر مختلف اصلی وجود دارد که عبارتند از :

۱. نوع موجودیت[۲۵]
۲. کلید اصلی[۲۶]
۳. پارامتر نوع موجودیت[۲۷]
۴. ارتباط[۲۸]

برای مثال برخی ویژگی‌های نوع موجودیت عبارتند از :

۱. نام مفرد
۲. نام جمع
۳. قابلیت حذف به صورت منطقی
۴. ...

در اینجا به ازای هر عنصر یک کلیشه تعریف شد و سپس به ازای هر ویژگی آن، یک برچسب مقداردار در کلیشه مربوطه طراحی شده است. همچنین نرم‌افزار Visual Paradigm امکان استخراج[۲۹] و وارد کردن[۳۰] کلیشه‌ها را در قالب فایل‌های xml فراهم می‌آورد. با این امکان می‌توان موارد طراحی شده را در اختیار مدلسازان پروژه‌ها قرار داد.

از آنجایی که ویژگی خواندن نمودار کلاس در خط تولید نرم‌افزار وجود داشت، پس از آشنایی با آن، به خواندن کلیشه‌های اعمال شده بر هر عنصر در نمودار کلاس می‌پرداختم. کلاس‌های کتابخانه emf که برای نمودار کلاس به زبان UML مورد استفاده قرار می‌گیرند، دارای یک تابع به نام getAppliedStereotypes هستند. این تابع فهرست کلیشه‌های اعمال شده به یک عنصر را برمی‌گرداند. سپس به کمک تابع getValue، می‌توان مقدار یک برچسب در یک کلیشه را از یک عنصر خواند.

پس از این مرحله به طراحی و پیاده‌سازی کلاس‌هایی به زبان جاوا پرداختم که تمامی اطلاعات مورد نیاز در نمودار کلاس، از جمله خود عناصر و کلیشه‌های اعمال شده به آن‌ها به همراه برچسب‌هایشان) دربردارد. پس از طراحی این کلاس‌ها، پس استخراج اطلاعات نمودار کلاس به کمک کتابخانه emf، آن اطلاعات را در ساختار کلاس‌های طراحی شده توسط خودم می‌ریختم و به کمک آن، مدل داده‌ای میانی در قالب فایل xml ساخته می‌شد.

---

۲۵ Entity
۲۶ Primary Key
۲۷ Property
۲۸ Association
۲۹ Export
۳۰ Import

## ۴.۴- ساخت پروژه از پروژه قالب با خواندن نمودار مؤلفه

در این بخش از کارآموزی به افزودن یک ویژگی جدید به خط تولید پرداختم. همانطور که در قسمت کار با Maven دیدیم، امکان ساخت یک پروژه از روی یک پروژه قالب با افزونه generate به سادگی امکانپذیر می‌باشد. در اینجا قصد داشتیم اطلاعات پروژه هایی را که قرار است از روی پروژه‌های قالب ساخته شود، از نمودار مؤلفه خوانده شود؛ بدین گونه که هر مؤلفه نمایانگر یک پروژه باشد. همچنین قرار شد اطلاعات پروژه به صورت برچسب‌های کلیشه بر هر مؤلفه افزوده گردد. لذا نیاز بود یک کلیشه به همراه برچسب‌های مورد نیازش طراحی و تعریف گردد (مشابه نمودار کلاس).

این ویژگی قرار بود در قالب یک افزونه‌ی maven پیاده‌سازی شود. یک افزونه برای ابزار maven، یک کلاس به زبان جاوا می‌باشد که کلاس AbstractMojo را گسترش می‌دهد. این کلاس می‌بایستی تابع execute را پیاده‌سازی کند. با اجرای افزونه، این تابع از کلاس فراخوانی می‌شود. همچنین برای گرفتن پارامترهای ورودی مورد نیاز افزونه، می‌بایستی به ازای هر پارامتر، یک متغیر با تفسیر[۳۱] Parameter در کلاس تعریف نمود. برای مثال :

```
@Parameter(property="umlBasedir" , defaultValue=".")
private String umlBasedir;
```

در این مثال یک پارامتر به نام umlBasedir از جنس رشته تعریف شده است. این پارامتر دارای یک مقدار پیش‌فرض نیز است.

در اینجا با نوشتن کلاس مورد نظر، ویژگی مورد نظر در قالب یک افزونه برای maven پیاده‌سازی شد. این کلاس دارای پارامترهای زیر برای گرفتن فایل نمودار مؤلفه و دیگر موارد است :

```
@Parameter(property="mavenHome" , defaultValue="${M2_HOME}")
private String mavenHome;

@Parameter(property="componentDiagrams" , required=true)
private String componentDiagrams;

@Parameter(property="destinationBasedir" , defaultValue="..")
private String destinationBasedir;

@Parameter(property="umlBasedir" , defaultValue=".")
private String umlBasedir;
```

این کلاس با خواندن نمودار مؤلفه مشخص شده و کلیشه‌های اعمال شده بر هر مؤلفه به همراه برچسب‌های

---

۳۱ Annotation

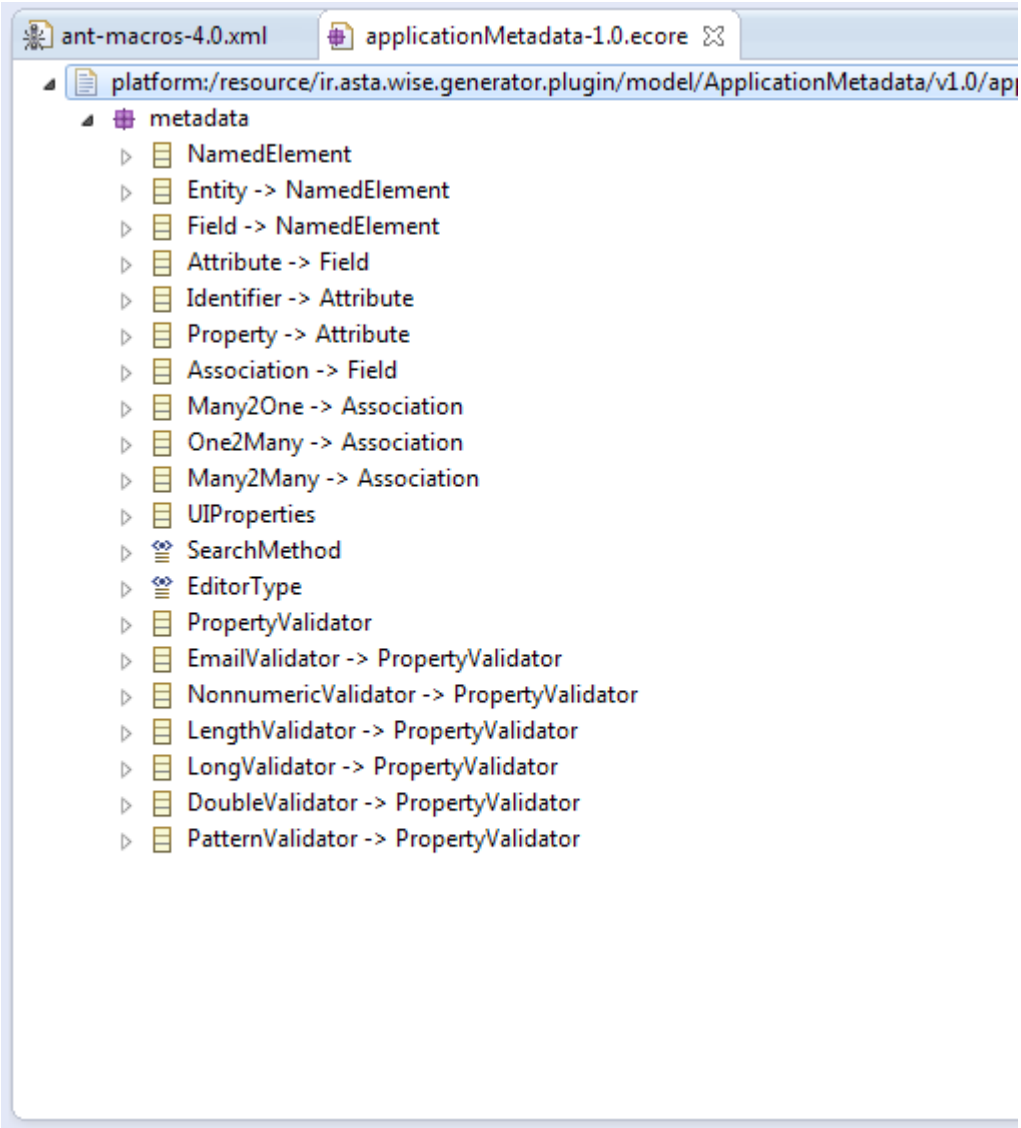
مقداردارشان، با اجرای افزونه generate از روی پروژه قالب مشخص شده با برچسب‌های کلیشه، یک پروژه می‌سازد.

## ۴.۵- بهبود ساخت خودکار کد از روی مدل داده‌ای میانی

در این مرحله به بهبود بخش ساخت خودکار کد، از روی مدل داده‌ای میانی، در خط تولید نرم‌افزار پرداختم. در حالت پیشین، مدل داده‌ای میانی، که در قالب یک فایل xml بود، در خط تولید خوانده می‌شد و با بازکردن عناصر درونی آن، کد کلاس‌های جاوا (که عمدتا کلاس‌های مربوط نوع موجودیت‌ها براساس تکنولوژی hibernate بود) ساخته می‌شد. اما با توجه به اینکه خواندن مستقیم عناصر فایل xml مربوط به مدل‌های داده‌ای و ساخت کد از روی آن‌ها، کد موجود انعطاف پذیری بسیار پایین داشت.

برای بهبود این بخش از خط تولید نرم‌افزار، ابتدا یک متامدل از روی این مدل داده‌ای ساخته شده. این متامدل شامل عناصر و ویژگی‌های استفاده شده در مدل داده‌ای میانی می‌باشد. پس از طراحی این متامدل، متامدل مورد نظر به کمک چارچوب مدلسازی Emf و متامدل ECore پیاده‌سازی شد. سپس به کمک این چارچوب مدلسازی، کلاس‌های مربوط به متامدل به صورت خودکار ساخته شد. سپس به پیاده‌سازی خواندن فایل‌های xml مدل داده‌ای و ریختن آن‌ها درون کلاس‌های متامدل پرداختم. این کار به کمک زبان groovy صورت گرفت. علت این امر هم سادگی خواندن عناصر فایل xml در این زبان و کتابخانه مربوطه‌اش می‌باشد. تصویر ۱ نمایی از این متامدل پیاده‌سازی شده را نمایش می‌دهد.

در ادامه کافیست به تغییر کد مربوط به ساخت کدهای کلاس‌ها با توجه به ساختار جدید پرداخت.




تصویر ۱: متامدل پیاده‌سازی شده به کمک *ECore*

# ۵- نتیجه‌گیری

با توجه به اینکه موضوع اصلی کارآموزی من، تکمیل بخشی از خط تولید نرم‌افزار بوده است، شاید یکی از مهم‌ترین دست‌آوردهای کارآموزی خود را آشنایی با خط تولید نرم‌افزار بدانم. خط تولید نرم‌افزاری که در شرکت اعوان استفاده می‌شود، نمودی از ساخت نرم‌افزار به عنوان یک محصول مهندسی با رویکردهای مهندسی است.

دیگر دست‌آورد مهم کارآموزی من آشنایی بیشتر با رویکرد ساخت و ایجاد نرم‌افزار مبتنی بر مدل بوده است. هرچند پیش از این درباره این راهکار، اطلاعاتی کلی (و نه دقیق) داشتم، کارآموزی در شرکت اعوان و کار در این حوزه، باعث آشنایی بیشتر من با این حوزه به صورت کاربردی و عملی شد. این آشنایی عملی باعث شده است نگرش من را درباره ایجاد نرم‌افزار تغییر دهد و سعی کنم در آینده از این رویکرد در برنامه‌سازی و ساخت نرم‌افزار بیشتر بهره جویم.

افزون بر موارد فوق، آشنایی با تکنولوژی Maven، که از تکنولوژی‌های رایج در صنعت نرم‌افزار است، برای من بسیار سودمند بوده است. Maven کاربردهای فراوانی دارد که می‌تواند بخشی از کارهای فرایند ساخت نرم‌افزار را، همچون کامپایل، آزمودن، تنظیم وابستگی‌ها، نصب، و دیگر موراد، به آن سپرد.

در انتها نیز از آقای پیشوایی به عنوان مسئول اصلی کارآموزی بسیار سپاسگزارم؛ زیرا تمامی موارد فوق با توجه به راهنمایی‌های ایشان صورت گرفته است. علاوه بر ایده‌های کاری که ایشان برخی از آن‌ها را به عنوان کارآموزی به من سپردند، در طی فرایند کاراموزی ایشان نکات فنی و صنعتی بسیاری را به من آموختند؛ به ویژه در مواردی که بین چند راه حل گوناگون شک داشتم و ایشان بهترین راه حل را پیشنهاد می‌کردند. حتی در برخی موارد که راه حل مناسبی نداشتم، ایشان راه حل‌هایی بسیار کارآمد ارائه می‌دادند و آشنایی با اینگونه راه حل‌ها بسیار برای من سودمند بوده است؛ زیرا به گفته خود ایشان کار در صنعت همچون حل مساله است، و حل مساله ذهن انسان را خلاق‌تر می‌کند.