

BSA Kullanıcı Kılavuzu

YGDYWiki sitesinden

Konu başlıkları

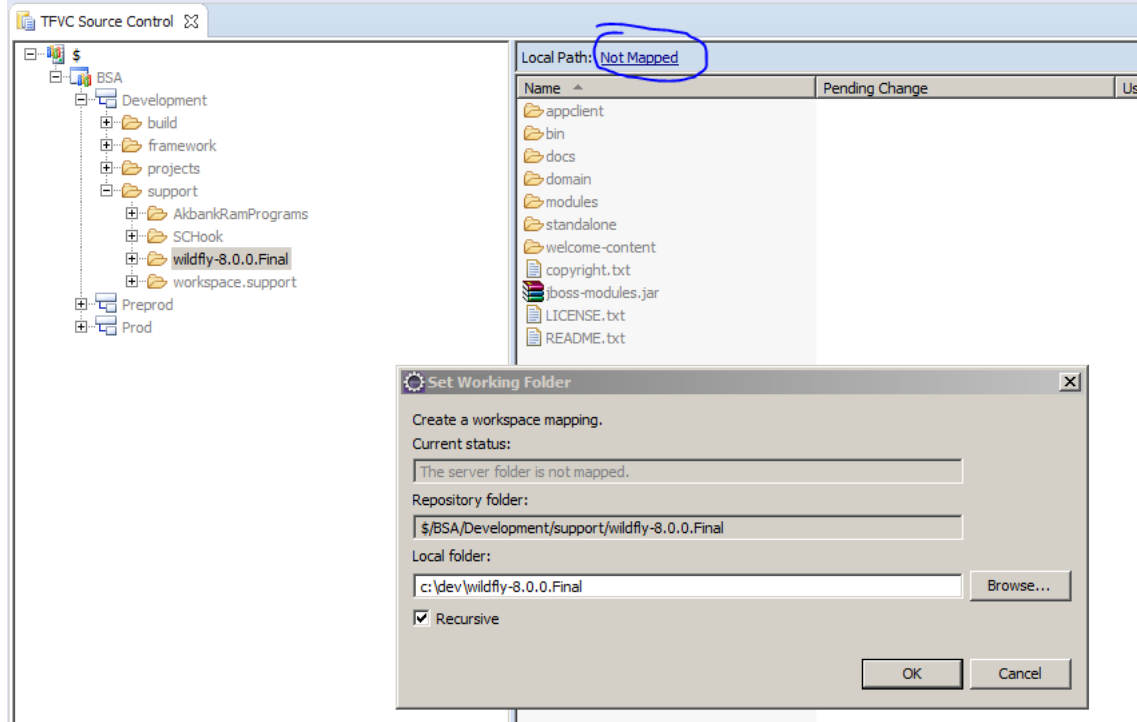
- 1 Başlangıç
 - 1.1 Ortam Kurulumu
 - 1.2 Yeni BSA Projesi Yaratma
 - 1.3 BSA Projesi package isimleri
 - 1.4 Projenin deploy edilmesi
 - 1.5 Yeni BSA Servis tanımlama
 - 1.6 BSA Servisinin Test Edilmesi
- 2 Error Codes&Messages
- 3 Exception Handling (BSAException)
- 4 Loglama
 - 4.1 MQ üzerinden asenkron loglama
- 5 Bag Usage Examples
- 6 Annotations
 - 6.1 @Service
 - 6.2 @Inbound
 - 6.3 @External
 - 6.4 @OnFailure
 - 6.5 @Scheduled
- 7 Batch
 - 7.1 Batch Tanımı
 - 7.2 Batch Durum Kontrolü
- 8 MyBatis
 - 8.1 Standart Tablo Kolonları
 - 8.2 MyBatis Generator Kullanımı
 - 8.3 Üretilen MyBatis Kodlarının Örnek Kullanımı
 - 8.4 Manuel MyBatis Sorguları Hazırlamak
 - 8.5 @Historical
- 9 MQ
 - 9.1 Yeni Queue ve Queue Connection Tanımı
 - 9.2 @MqConsumer
 - 9.2.1 @MqConsumer Attributes
 - 9.3 Mesaj Göndermek
- 10 Web Servis
 - 10.1 MCA Artifacts ve Wsdl oluşturma
 - 10.2 Web Servis çağırma
 - 10.3 XSD'den Jaxb Class'ları oluşturma
- 11 Extending BSACache
- 12 Reference Data
 - 12.1 Defining new Reference Data
 - 12.2 Retrieving Data from Reference Data
 - 12.3 BSA_LOOKUP_DATA tanımlama
- 13 Calling Service With ServiceExecutor
- 14 ConfigurationHelper kullanımı
- 15 BSAContext
- 16 Db Authorization
- 17 Service Authorization
- 18 Adding Third Party Libraries
- 19 Kod Review Check List
 - 19.1 Java Standards
 - 19.1.1 Project Dependency
 - 19.1.2 Servis isimlendirmeleri
 - 19.1.3 Servis Javadoc
 - 19.1.4 Fiziksel Delete
 - 19.1.5 DAO katmanı
 - 19.1.6 Try\Catch
 - 19.1.7 SystemPrintln
 - 19.1.8 Metod uzunluğu
 - 19.1.9 Metod parametre sayısı
 - 19.1.10 Mapper Factory referans
 - 19.1.11 Servis package
 - 19.1.12 Package isimlendirmeleri
 - 19.1.13 Generic Exception catch etme
 - 19.1.14 Throwable catch etme
- 20 BSA Ortam bilgileri
 - 20.1 Uygulama sunucuları
 - 20.1.1 Veritabanı kullanıcıları
 - 20.1.2 Datasource'ların kullandığı kullanıcılar :
- 21 Deployment işlemleri
- 22 Deployment Sonrası Kontroller
- 23 WhitePlugin
 - 23.1 Güncelleme
- 24 BSA Server'larında çalışan uygulamalar
- 25 MCA - BSA servis ilişki tablosu

Başlangıç

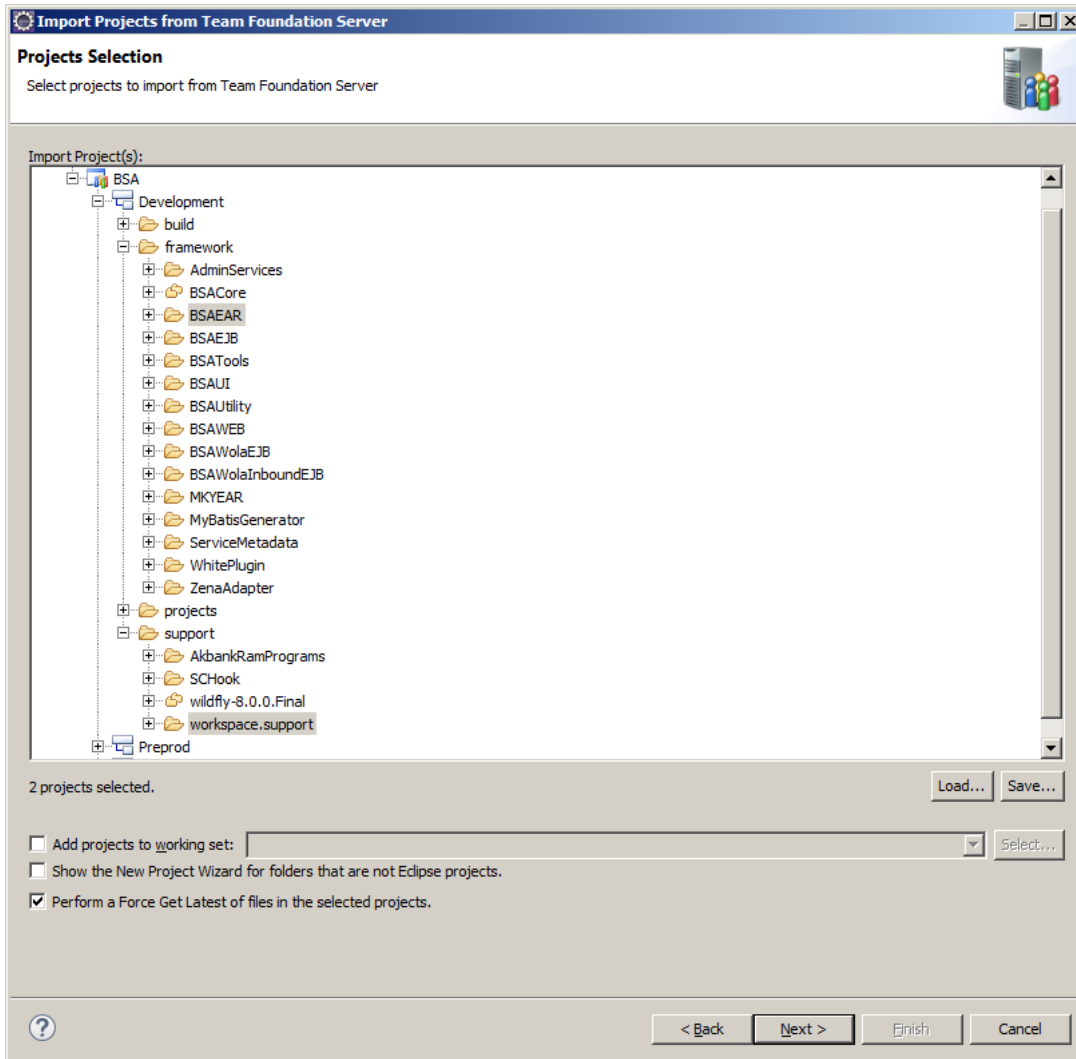
Ortam Kurulumu

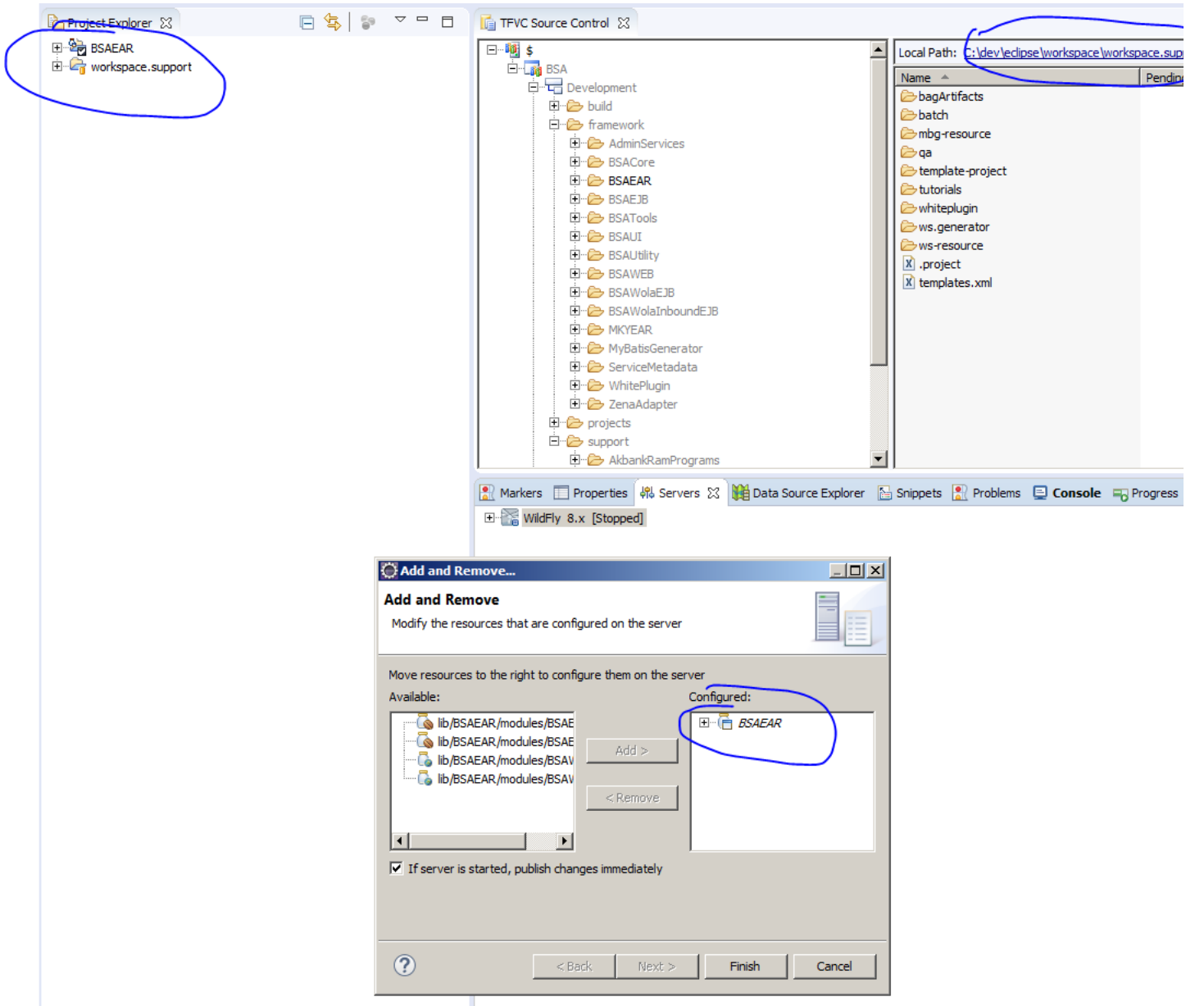
Kurulum adımları:

- jdk1.8 kurulmalıdır. (Projeler 1.7 ile compile edilmeli, 1.8 eclipse neon için gereklidir.) \\Sn000001\\sys\\Data\\Project\\2012Proj\\NewBackendArchitectureFramework\\ApplicationServiceFramework\\install files altında mevcut.
- \\Sn000001\\sys\\Data\\Project\\2012Proj\\NewBackendArchitectureFramework\\ApplicationServiceFramework\\install files folder'ı altından eclipse-jee-neon-1a-win32-x86_64_BSA.zip (Local admin yetkisi yoksa zip dosyası c:\\temp altına kopyalanması şiddetle tavsiye edilir.) c:\\dev\\eclipse altına extract edilir.
- Eclipse'de Team Foundation Server Exploring perspective'inden (TFS sunucusu : tfssrv2) TFS'deki \$/BSA/Development/support/wildfly-8.0.0.Final c:\\dev\\wildfly-8.0.0.Final altına maplenir. Bu şekilde JBoss WildFly 8.0 sunucusu lokal makineye indirilmiş olur.



- Eclipse'de Project Explorer'da sağ tıklayarak menüdeki import açılır. Açılan ekranda Team'Projects from''''Team Foundation Server seçilir next ile ilerlenir. \$/BSA/Development/framework/BSAEAR ve \$/BSA/Development/support/workspace.support ekran görüntüsündeki gibi seçilerek workspace'e eklenir. Not: import edilen projelerin c:\\dev\\eclipse\\workspace altında olması gereklidir.





- Eclipse'de server view'ındaki WildFly 8.0 a sağ tıklanarak **Add/Remove Projects** penceresi açılarak BSAEAR eklenir. Server clean edildikten sonra debug modunda açılır.

Eğer mevcut bir BSA projesi üzerinde çalışılacaksa bu proje içeriğini görmek için TFS yetkisi istenmelidir. Bu işlem için önce, Aksiyon üzerinden değişiklik talebi açılır Kategori 3 olarak TFS seçilir (Aksiyon\Değişiklik\Yazılım Ürünleri\Güvenlik\Tfs) . Yeni açılan Görev tabındaki combobox'tan "TFS Proje ve Yetki Talepleri" seçildikten sonra form ekranına aşağıdaki bilgiler girilir. Kullanıcı Domain User olarak Active Directory sicilinizi girmeniz gerekmektedir. Bu siciller N ile başlar. Service'ler kullanıcı gruplarına göre yetkilendirilmiştir (Framework (core) serviceleri tüm gruplar için ortaktır, aşağıdaki listeden size uygun olanı Role alanında LOB seçtikten sonra Lob Adı bölümüne yazmanız gerekmektedir. Lob adı "Component-project developers" yapısındadır.

Örnek Aksiyon kaydı aşağıda görülebilir

Kategori 1	Kategori 2	Kategori 3	Kategori 4

 Kategorilerde Ara

Seçili Görevi Sil

Görev
GF-2016-299924 TFS Proje ve Yetki Talepleri

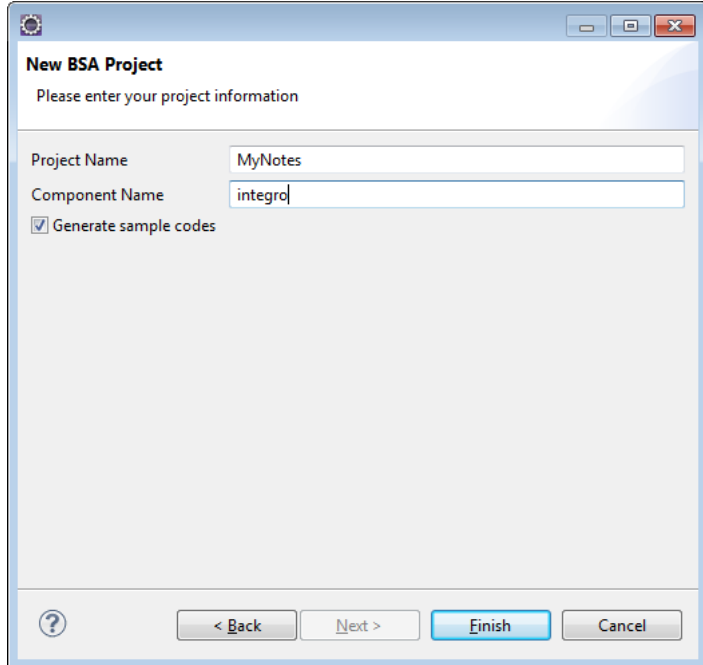
TFS Proje ve Yetki Talepleri

Form	TFS Kullanıcı Yetkilendirme Formu
Server	<input type="radio"/> TFS 2008 <input type="radio"/> TFS 2010 <input checked="" type="radio"/> TFS 2015 defaultcollection
TeamProject	BSA
Kullanıcı Domain User	n56635
Role	<input type="radio"/> Reader(R) <input type="radio"/> Contributor(RW) <input checked="" type="radio"/> LOB
TFS Repository Dizinleri için bkz: http://tfssrv01:8080/wiki/index.php/Resim:BranchBanking_Folder.png	
Lob Adı	core-Admin Developers
Açıklama	Bsa geliştirmesi yapabilmesi için ilgili LOB grubuna kullanıcının eklenn
Bağlı Görev	[...]

Açıklamalar :

Yeni BSA Projesi Yaratma

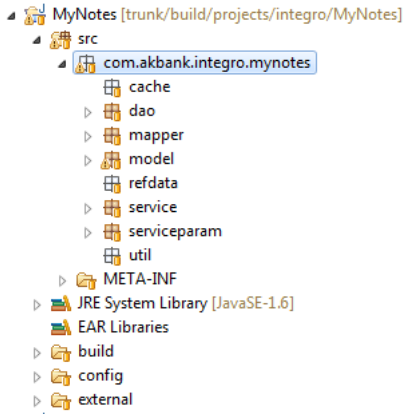
Eclipse üzerinde menüden File\New\Other\New BSA Project seçilerek Yeni BSA Projesi yaratma ekranı açılır. Proje ve component isimleri girilir, finish butonuna basılır. BSA projeleri yaratılırken projenin ilişkili olduğu ana bileşene göre sınıflandırılarak Component name alanı girilir. Proje yaratılırken package isimleri com.akbank.[component.name].[project.name] altında yaratılır.



BSA Projesi package isimleri

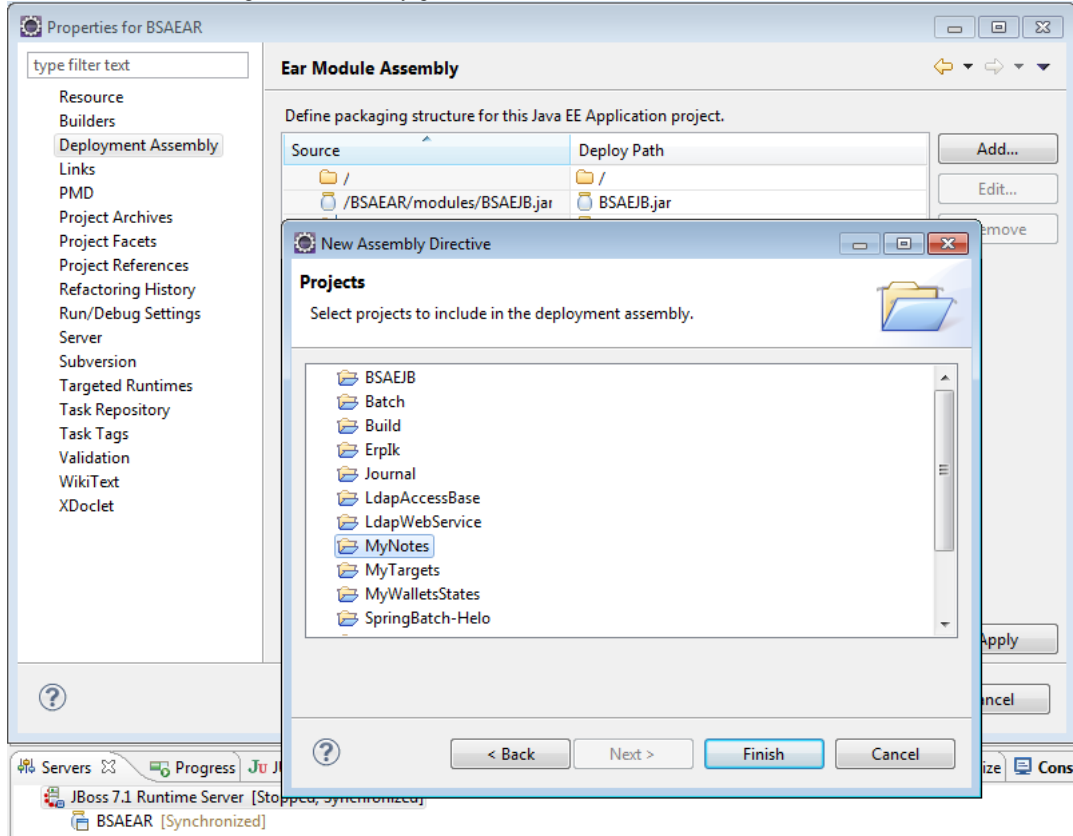
BSA Projesi yaratıldığında aşağıdaki package'lar otomatik olarak yaratılmaktadır. İhtiyaç duyulması durumunda bu package'ların dışında yeni package'lar developer tarafından yaratılabilir.

- cache** : Projeye özel cache'de veri tutma ihtiyacı olması durumunda class'lar cache package altında yaratılır.
- dao** : Veritabanına erişmek için dao package'ında static metotlar yazılır.
- mapper** : MyBatis Generator tarafından üretilen ve custom query yazılması gerektiğinde mapper.xml ve mapper.java dosyaları mapper package'ında yaratılır.
- model** : Veritabanındaki tablolara karşılık gelen model class'ları model package'ı altında MyBatis Generator tarafından yaratılmaktadır.
- refdata** : Projeye özel Referans data ihtiyacı olması durumunda Referans Data loader class'lar refdata package'ı altında yaratılmaktadır.
- service** : BSA servislerinin geliştirileceği class'lar service package'ı altında yaratılır.
- serviceparam** : Dışarıya açılacak servislerin input/output parametrelerinin bulunduğu class'lar serviceparam package'ı altında yaratılır.



Projenin deploy edilmesi

BSAEAR projesi zerin saę tıklanarak properties ekranında Deployment Assembly seçilerek proje eklenir. Proje BSAEAR'a eklendikten sonra Servers\JBoss 7.1 Runtime Server üzerinde saę tıklanarak clean yapılır.



Yeni BSA Servis tanımlama

BSA servisleri servis class'larında aşağıdaki şekilde public static metod'lar yaratılarak tanımlanır. Tanımlanan servis ServiceExecutor.execute("INTEGRO_MYNOTES_FETCH_NOTES") metodu ile başka servislerden çağrılabilir. Servis class'ları direkt servis package'ı altında oluşturulmalıdır.

```
@Service("INTEGRO_MYNOTES_FETCH_NOTES")
public static Bag fetchNotes(Bag inBag) throws BSAException {
    Bag outBag = new Bag();

    return outBag;
}
```

BSA Servisinin Test Edilmesi

BSA servisleri Junit ile test edilebilir. Tester class'ının execute metodu kullanılarak BSA servisi çalıştırılabilir. Tester.execute metodu Bag veya Object array'i alabilir. Object olarak @Inbound servis'in argümanları sırası ile parametre olarak geçilmelidir. Dışarıya açık servisler (@Inbound) soup-ui ile de test edilebilir.

```
public class ProductServiceTester {
    private final static Logger log = Logger.getLogger(ProductServiceTester.class);

    @Test
    public void selectAllProductsPlugin() throws BSAException {
        Bag inBag = new Bag();
        Bag outBag = Tester.execute("INFRA_PRODUCT_SELECT_ALL_PRODUCTS_PLUGIN", inBag);
        log.info(outBag);
    }
}
```

```

    }

    @Test
    public void testBatchInsert() throws BSAException {
        List<Product> products = new ArrayList<Product>();
        for (int i = 0; i < 5; i++) {
            Product product = new Product();
            product.setName("Test product " + i);
            product.setValue(new BigDecimal(10 * i));
            product.setProductDate(new Date());
            product.setIsEnabled(true);
            products.add(product);
        }
        Bag inBag = CoreUtil.listToBag(products, "PRODUCT_LIST");
        Tester.execute("INFRA_PRODUCT_BATCH_INSERT", inBag);
    }

    @Test
    public void testSelectCustomValuesIn() throws BSAException {
        Bag inBag = new Bag();
        List<BigDecimal> values = new ArrayList<BigDecimal>();
        values.add(new BigDecimal(1599));
        values.add(new BigDecimal(1));
        values.add(new BigDecimal(899));

        for (int i = 0; i < values.size(); i++) {
            inBag.put("VALUE_LIST", i, "VALUE", values.get(i));
        }

        Bag outBag = Tester.execute("INFRA_PRODUCT_SELECT_CUSTOM_VALUES_IN", inBag);
        log.info(outBag);
    }
}

```

Error Codes&Messages

Hata kodları ve açıklamaları BSA_ERROR tablosuna girilmektedir. ErrorCodes.get(String errorCode,Object ... arguments) metodu ile hata açıklamasına ulaşabilir.

```

ErrorCodes.get("IAM_001");

Hata açıklama örnek : Lütfen kullanıcı numarasını giriniz : {0}

ErrorCodes.get("IAM_002",userNumber);

```

Exception Handling (BSAException)

Throw edilen Exception'lar framework tarafından yakalanmakta, BSA servislerini çağıran dış sistemlerin alabileceği formatta response oluşturulmakta ve hata log mesajları üretilmektedir. BSA servisleri içerisinde try/catch blokları yazılmamalıdır. İş kurallarına göre hata fırlatılması gerekiyorsa BSAException throw edilmelidir. BSAException objesi yaratırken hata açıklamaları ErrorCodes class'ı kullanılarak oluşturulmalıdır

```

User user = UserDao.retrieveUser(appId, registrationNumber);
if (user==null){
    throw new BSAException(ErrorCodeConstants.ERROR_CODE_USER_NOT_EXISTS);
}

```

Sistemsel hatalar olması durumunda framework tarafından BSARuntimeException nesneleri throw edilmektedir.

Loglama

BSA uygulaması tarafından bilgi, hata ve performans logları atılmaktadır. Loglar lokal ortamda System.out çıktılarının yazıldığı konsola atılacak şekilde, sunucular üzerinde ise \weblog\applog\[JVM ismi] klasöründe dosyaya atılacak şekilde konfigüre edilmiştir. Websphere uygulama sunucusu tarafından atılan SystemOut ve SystemErr logları ise \weblog\waslog\[JVM ismi] klasöründe bulunmaktadır.

Örneğin UAT ortamına hizmet veren sunucunun uygulama logları /weblog/applog/BSAUAT klasöründe server logları ise /weblog/waslog/BSAUATSrv1 ve /weblog/waslog/BSAUATSrv2 klasörlerinde bulunmaktadır.

Üretim ortamında da Integro sunucusunun uygulama logları /weblog/applog/BSAIntegroSrv1 klasöründe server logları /weblog/waslog/BSAIntegroSrv1 bulunmaktadır.

Test ortamlarında log'lara erişmek için "bsalog" kullanıcısı ile sunucuya login olunabilir; bu kullanıcının şifresi "bsalog123" tür. Log'ların bulunduğu /weblog/applog ve /weblog/waslog dizinleri için gerekli yetkiler bu kullanıcıya verilmiştir.

- **Hata logları** aşağıdaki yapıda atılmaktadır. Uygulamalar tarafında alınan hata dosyasının default service-error olarak isimlendirilmiştir. Log dosyaları ihtiyaç duyulması durumunda component bazında ayrıştırılabilmektedir. Ayrı log dosyasına ihtiyaç duyulması durumunda BSA altyapı ekibi ile temasa geçilmelidir. Eğer component ismine özel tanım yapılmaz ise hata logları ortak error.log dosyasına atılmaktadır.

```

<error
  request id="adf66d40-7d6e-4c94-877a-e0224de2b2af0010011272R"
  error code="INT_003"
  channel="010101"
  user="48754"
  timestamp="2013.08.16 AD at 23:59:40 EEST"
  thread="WebContainer : 8">
</message>
<message>
  Error parsing date : 1
</message>
<throwable>
<![CDATA[com.akbank.bsa.core.exception.BSAException: Error parsing date : 1
  at com.akbank.bsa.core.utils.DateUtil.getDate(DateUtil.java:37)
  at com.akbank.integro.mytargets.service.MyTargetsService.getSalesTargetInfo(MyTargetsService.java:76)
  at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

```

```
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:60)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:37)
at java.lang.reflect.Method.invoke(Method.java:611)
at com.akbank.bsa.core.ServiceExecutor.execute(ServiceExecutor.java:77)
at com.akbank.bsa.service.ejb.EJBGatewayServiceBean.processRequest(EJBGatewayServiceBean.java:68)
at com.akbank.bsa.service.ejb.EJBGatewayServiceBean.processRequestObjectNewTx(EJBGatewayServiceBean.java:53)
at com.akbank.bsa.service.ejb.EJBRemote05LEJBGatewayServiceCMT_31304823.processRequestObjectNewTx(EJBRemote05LEJBGatewayServiceCMT_31304823.java)
at com.akbank.bsa.service.ejb._EJBGatewayService_Stub.processRequestObjectNewTx(_EJBGatewayService_Stub.java)
at com.akbank.bsa.core.ServiceExecutor.executeInNewTx(ServiceExecutor.java:124)
at com.akbank.bsa.ws.BSAWebService.call(BSAWebService.java:85)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:60)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:37)
at java.lang.reflect.Method.invoke(Method.java:611)
at org.apache.axis2.jaxws.server.dispatcher.JavaDispatcher.invokeTargetOperation(JavaDispatcher.java:113)
at org.apache.axis2.jaxws.server.dispatcher.JavaBeanDispatcher.invoke(JavaBeanDispatcher.java:118)
at org.apache.axis2.jaxws.server.EndpointController.invoke(EndpointController.java:111)
at org.apache.axis2.jaxws.server.JAXWSMessageReceiver.receive(JAXWSMessageReceiver.java:161)
at org.apache.axis2.engine.AxisEngine.receive(AxisEngine.java:208)
at org.apache.axis2.transport.http.HTTPTransportUtils.processHTTPPostRequest(HTTPTransportUtils.java:172)
at com.ibm.ws.websvcs.transport.http.WASAxis2Servlet.doPost(WASAxis2Servlet.java:1532)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:595)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:668)
at com.ibm.ws.webcontainer.servlet.ServletWrapper.service(ServletWrapper.java:1225)
at com.ibm.ws.webcontainer.servlet.ServletWrapper.handleRequest(ServletWrapper.java:775)
at com.ibm.ws.webcontainer.servlet.ServletWrapper.handleRequest(ServletWrapper.java:457)
at com.ibm.ws.axis2.jaxws.servlet.ServletWrapperImpl.handleRequest(ServletWrapperImpl.java:178)
at com.ibm.ws.webcontainer.filter.WebAppFilterManager.invokeFilters(WebAppFilterManager.java:1032)
at com.ibm.ws.webcontainer.servlet.CacheServletWrapper.handleRequest(CacheServletWrapper.java:87)
at com.ibm.ws.webcontainer.WebContainer.handleRequest(WebContainer.java:908)
at com.ibm.ws.webcontainer.WSWebContainer.handleRequest(WSWebContainer.java:1662)
at com.ibm.ws.webcontainer.channel.WCChannelLink.ready(WCChannelLink.java:195)
at com.ibm.ws.http.channel.inbound.impl.HttpInboundLink.handleDiscrimination(HttpInboundLink.java:453)
at com.ibm.ws.http.channel.inbound.impl.HttpInboundLink.handleNewRequest(HttpInboundLink.java:515)
at com.ibm.ws.http.channel.inbound.impl.HttpInboundLink.processRequest(HttpInboundLink.java:306)
at com.ibm.ws.http.channel.inbound.impl.HttpInboundLink.ready(HttpInboundLink.java:277)
at com.ibm.ws.tcp.channel.impl.NewConnectionInitialReadCallback.sendToDiscriminators(NewConnectionInitialReadCallback.java:214)
at com.ibm.ws.tcp.channel.impl.NewConnectionInitialReadCallback.complete(NewConnectionInitialReadCallback.java:113)
at com.ibm.ws.tcp.channel.impl.AioReadCompletionListener.futureCompleted(AioReadCompletionListener.java:166)
at com.ibm.io.async.AbstractAsyncFuture.invokeCallback(AbstractAsyncFuture.java:217)
at com.ibm.io.async.AsyncChannelFuture.fireCompletionActions(AsyncChannelFuture.java:161)
at com.ibm.io.async.AsyncFuture.completed(AsyncFuture.java:138)
at com.ibm.io.async.ResultHandler.complete(ResultHandler.java:204)
at com.ibm.io.async.ResultHandler.runEventProcessingLoop(ResultHandler.java:816)
at com.ibm.io.async.ResultHandler$2.run(ResultHandler.java:905)
at com.ibm.ws.util.ThreadPool$Worker.run(ThreadPool.java:1691)
]]>
</throwable>
</error>
```

- **Performans logları** tree şeklinde comma seperated olarak ortalama değer hesaplama amaçlı olarak iki şekilde atılmaktadır. Tree şekilde atılan performans loglarında her bir servisin ve çağırdığı servisin çalışma süreleri milisanıye bazında gösterilmektedir.

```
Performance Thread[BSA Scheduler_Worker-1] 2013-08-22 13:13:42,004
|->BATCH_PROCESS_SERVICE 117988
|  |->BATCH_CHILD_STATE_SELECT_FOR_UPDATE 287
|  |->INFRA_AUTHORIZATION_SAP_BATCH_SERVICE 117544
|     |->INFRA_AUTHORIZATION_SAP_FETCH_EMPLOYEES 96193
|     |->INFRA_AUTHORIZATION_SAP_FETCH_JOB_LEVELS 618
|     |->INFRA_AUTHORIZATION_SAP_FETCH_ORGANIZATION 20437
|     |->INFRA_AUTHORIZATION_BATCH_INSERT_EMPLOYEE 126
|     |->INFRA_AUTHORIZATION_BATCH_UPDATE_EMPLOYEE 109
|     |->INFRA_AUTHORIZATION_BATCH_DELETE_EMPLOYEE 60
|  |->BATCH_STATE_SELECT_FOR_UPDATE 31
|  |->BATCH_CHILD_STATE_UPDATE 7
|  |->BATCH_CHILD_STATE_GET_BATCH_STATE 69
|  |->BATCH_STATE_UPDATE 41
```

Comma seperated olarak oluşturulan performans loglarından BSA altyapı ekibi tarafından geliştirilen extractPerformanceReport.jar kullanılarak ortalama servislerin ortalama çalışma süreleri hesaplanabilir.

```
2013-08-22 13:13:41,555;INFRA_AUTHORIZATION_SAP_FETCH_ORGANIZATION;20437
2013-08-22 13:13:41,681;INFRA_AUTHORIZATION_BATCH_INSERT_EMPLOYEE;126
2013-08-22 13:13:41,791;INFRA_AUTHORIZATION_BATCH_UPDATE_EMPLOYEE;109
2013-08-22 13:13:41,851;INFRA_AUTHORIZATION_BATCH_DELETE_EMPLOYEE;60
2013-08-22 13:13:41,851;INFRA_AUTHORIZATION_SAP_BATCH_SERVICE;117544
2013-08-22 13:13:41,887;BATCH_STATE_SELECT_FOR_UPDATE;31
2013-08-22 13:13:41,893;BATCH_CHILD_STATE_UPDATE;6
2013-08-22 13:13:41,963;BATCH_CHILD_STATE_GET_BATCH_STATE;69
2013-08-22 13:13:42,004;BATCH_STATE_UPDATE;41
2013-08-22 13:13:42,004;BATCH_PROCESS_SERVICE;117996
```

- **info logları** standard.log dosyasına atılmaktadır.

MQ üzerinden asenkron loglama

Uygulamalara özel loglama ihtiyaçları asenkron olarak MQ üzerinden loglanmaktadır. Loglama için aşağıdaki adımlar izlenmelidir :

- BSA_QUEUE_CONNECTION_DEF ve BSA_QUEUE_DEFINITION tablolarına Queue bilgileri girilmelidir. BSA_QUEUE_CONNECTION_DEF tablosunda bağlanacağımız queue manager tablosunda CLASS_NAME kolonuna BSAArchitects ekibi tarafından class'ın ismi girilmelidir. Örneğin : com.akbank.bsa.core.jms.cf.DocumentManagementQueueManager.
- MQ'ya log mesajı gönderecek BSA projesinde mq package'ı altında BSA_QUEUE_CONNECTION_DEF tablosundaki CLASS_NAME'i extend eden boş bir class yaratılmalıdır. Örneğin : public class DYSQueueManager extends DocumentManagementQueueManager
- BSAMqLogger.putObjecttoQueue kullanılarak log mesajı MQ'ya put edilir. Örnek kullanım : BSAMqLogger.putObjecttoQueue(DYSQueueManager.class, BSA_QUEUE_DEFINITION.ALIAS değeri, bag.expose())
- Loglama işlemini yapacak @MqConsumer bir BSA servisi geliştirilir.

Bag Usage Examples

Annotations

@Service

Servis metotlarının üzerine eklenir. İçerisine servis ismi yazılır.

```
@Service("INTEGRO_MYNOTES_FETCH_NOTES", allowedServices = { "INTEGRO_MYTARGETS_GET_TARGET_ARGUS_DATE", "INTEGRO_MYTARGETS_GET_TARGET_HR_LOAD_DATE"})
public static Bag fetchNotes(Bag inBag) throws BSAException {
    Bag outBag = new Bag();

    return outBag;
}
```

allowedServices attribute'u deprecated olmuştur. Servis yetkilendirme için db'de service authorization tablosu oluşturulmuştur ve servis yetkilendirme için bu tablo kullanılacaktır.

@Inbound

BSA servisinin inbound olarak çağırılabilmesi için @Inbound annotation'ı metoda eklenir. Object veya object listesi input olarak alabilir. Object return edebilir. Önemli olan input/output değişkenlerin xml'e serialize/deserialize edilebilir olmasıdır. Xml'e serialize/deserialize edilebilmesi için (primitive type'ların dışındakiler için) class'ın başına mutlaka @XmlRootElement annotation'ı eklenmelidir.

```
@Service(value="DEMO_DEMO_PROJECT_ONE_INBOUND_TEST")
@Inbound
public static boolean inboundTest(String deneme, ObjectRequest request) throws BSAException {

    log.info(request);

    return true;
}
```

```
@XmlRootElement
public class ObjectRequest {
    private File file;
    private Customer customer;
    private byte[] stream;
}
```

@External

Inbound olarak çağırılacak servislerin üzerine @External annotation tag'i eklenir ve bu tag içerisinde servisin input/output parametreleri belirlenir. Input/Output parametrelerinin bulunduğu class'lar serviceparam package'ında yaratılır.

```
@Service("INTEGRO_MYNOTES_FETCH_NOTES")
<strike>@External</strike>(inputClass = InputClassFetchNotes.class, outputClass = OutputClassFetchNotes.class)
public static Bag fetchNotes(Bag inBag) throws BSAException {
    Bag outBag = new Bag();
    String userName = inBag.get("USER_NAME").toString();
    Date noteDate = inBag.get("NOTE_DATE").toDate();
    boolean fetchAll = inBag.get("FETCH_ALL").toBoolean();

    ...
    ...
    int index = 0;
    for (Note note : notes ) {
        note.toBag(outBag, "NOTE_TABLE", index);
        index ++;
    }

    return outBag;
}
```

Servise input olarak gönderilecek değişken isimlerinin bulunduğu class. Service gönderilen inbag objesine key değerleri input class'ındaki değişken isimlerindeki büyük harflerin önüne "_" konularak uppercase olarak framework tarafından put edilir. Servis içerisinde bu değerlere inBag içerisinde yukarıdaki örnekte görüldüğü gibi erişilir.

```
public class InputClassFetchNotes {
    private String userName;
    private Date noteDate;
    private boolean fetchAll;
}
```

Servisin output olarak döneceği değişken isimlerinin bulunduğu class. Outbag objesine değerler put edilirken output class'ındaki değişken isimlerindeki büyük harflerin önüne "_" konularak uppercase olarak put edilmelidir. Outbag'a konulan değerler framework tarafından Soap mesajına konulur.

```
public class OutputClassFetchNotes {
    List<Note> noteTable;
}
```

@OnFailure

Bir servisin hata alması durumunda tüm veritabanı işlemleri rollback edilir. Ancak bazı durumlarda hata alınca veritabanı işlemleri rollback edilip normal akış devam ederken, yeni bir transaction açarak yapılması gereken işlemler olabilir. Bu ihtiyaç olması durumunda servis metoduna @OnFailure annotation tag'i eklenir. OnFailure annotation tag'ine hata alması durumunda çağrılacak servis ismi ve hangi hatalarda OnFailure servisinin çağrılacağı girilir.

```
@Service("INFRA_AUTHORIZATION_DELETE_USERS")
@OnFailure(value="INFRA_AUTHORIZATION_DELETE_USERS_FAILURE",types={SQLException.class})
public static Bag deleteUser(Bag inBag) throws BSAException {
    ...
    ...
    return new Bag();
}
```

@Scheduled

Bir servisin belirli zamanlarda tetiklenmesini sağlar. Parametre olarak cron expression ve disableConcurrentExecute alır. disableConcurrentExecute attribute'u optional'dır, default'u false'dur.

disableConcurrentExecute attribute'u verilmediğinde ya da değeri false olarak verildiğinde servis cron expression'da belirtilen zamanlarda tüm instance'larda çalışır. Bu attribute'un değeri true verildiğinde cron expression'da belirtilen zamanlarda sadece tek instance'da çalışır.

Örnek kod aşağıdaki gibidir.

```
@Service(value = "INFRA_AUTHORIZATION_RELOAD_USER_CACHE")
@Scheduled(cron="0 0 6 * * **", disableConcurrentExecute="true") //Hergün saat 06:00 da çalışır.
public static Bag reloadUserCache(Bag inBag) throws BSAException {
    return null;
}
```

BSA_CONFIG tablosuna schedule edilen servisin adı eklenmelidir. Test ortamlarında clusterName ALL olarak girilmelidir. Prod ortamda ise uygulama hangi cluster'da çalışacak ise o cluster'ın adı girilmelidir. Örnek kayıt aşağıdaki gibidir.

```
section:scheduledServices
tag:INFRA_AUTHORIZATION_RELOAD_USER_CACHE
value:true
clusterName:Integro
```

Cron expression syntax'ı ve örnekleri için : <http://www.quartz-scheduler.org/documentation/quartz-2.2.x/tutorials/crontrigger>

Batch

Batch Tanımı

Batch işlemler için Bag alıp Bag dönen internal bir BSA servisi geliştirilmeli ve aşağıdaki şekilde BSA_BATCH_DEFINITION tablosuna tanım yapılmalıdır. Batch işin birden fazla thread açılarak çalışması gerekirse THREAD_COUNT kolonuna açılacak thread sayısı girilmelidir. Batch servise THREAD_COUNT(Toplam thread sayısı) ve THREAD_NO(Kaçıncı thread olduğu) key değerleri gönderilmektedir. Bu bilgilerle veritabanından her thread'in işlemesi gereken kayıtlar çekilerek işlenebilir.

```
Insert into BSA_BATCH_DEFINITION
(ID, BATCH_NAME, BATCH_GROUP_NAME, DESCRIPTION, PROCESS_SERVICE, THREAD_COUNT, BATCH_TYPE, WHEN_EXCEPTION)
Values
('30', 'INFRA_AUTHORIZATION_ERROR_LOG_BATCH', 'AUTHORIZATION', 'Hatalı işlemleri tekrar edilmesi',
'INFRA_AUTHORIZATION_ERROR_LOG_BATCH', 1, 'DEFAULT', 'CONTINUE');
```

Üretim ortamında Zena'dan çağırılan batch işler "BSA batch" sunucusunda çalışmaktadır. Prod için parametre kısmında geçen adres aşağıdaki gibi olmalıdır.

<http://bsabatch.akbank.com.tr>

Batch servisin Zena tarafında schedule edilmesi için aşağıdaki şekilde Aksiyon kaydı girilmelidir.

[Kapat](#) [Değişikliği Aç](#) [Yazdır](#)

ZENA Tanımları

Statü : **Tamamlandı** Yetkili :

▼ Talepteki Diğer Görevler :

Kayıt Açanın Bilgileri			Kayıt T
Kayıt Açan		52219-MEHMET LEVENT TOKMAK	Talep
Birim		5171-BT KURUMSAL MİMARİ YÖNETİMİ	Ortam
Alt Grup		TMAG - TEKNOLOJİ MİMARİ ALT GR	Katego
Görev Tanımı			İdari (
İmza Yetkisi			İdari (
Talep Açıklama		TBSA0002 den sonra TBSA0003'den önce sırayla çalışacak şekilde ve aşağıdaki parametre ile zena tanımının yapılması. Parametre:" EXECUTE AUTHORIZATION INFRA_AUTHORIZATION_FEED_USER_RIGHTS_TO_SYSTEMS http://bsaentegre.akbank.com.tr:80"	
Talep Ek Dokümanları			
Görev Açıklama		TBSA0002 den sonra TBSA0003'den önce sırayla çalışacak şekilde ve aşağıdaki parametre ile zena tanımının yapılma Parametre:" EXECUTE AUTHORIZATION INFRA_AUTHORIZATION_FEED_USER_RIGHTS_TO_SYSTEMS http://bsaentegre.akbank.com.tr:80"	
Bu görevi bekleyen görevler			Talebe

İşlem Açıklaması		TBSAGPR! processinde PBSA0007 taskı oluşturuldu. Akış aşağıdaki gibi oldu. TBSAGPR1 processi --> TBSAG0002 --> TBSAG0007 --> TBSAG0003 --> TBSAG0005 --> TBSAG0006
Görev Ek Dokümanları		

Bu görev idari onaylı olup, idari onay için yöneticinize gidecektir. Görev Grubu İŞ PLANLAMA VE YÜRÜTME ALT grubudur.

Telefon	63124	Ortam
İşlem tipi	Tanım	Kritik yolda mı?
İşin adı	Aktifleşen expire eden r	İşletim sistemi
Script adı	zenaadapter.sh	Script path'i
ZEKE bağlantısı var mı?	Hayır	

Açıklamalar :

Test ortamı için aşağıdaki bilgilerle tanım yapılmalıdır.

Sunucu : akbatkont1 (172.21.130.60)

User: bsazena

Path: /home/users/bsazena

Script: zenaadapter.sh

Batch Durum Kontrolü

Batch işler her thread için BSA_BATCH_CHILD_STATE tablosuna kayıt insert edilir. Batch'ler aşağıdaki statüde olabilir.

- COMPLETED : Başarılı olarak tamamlanmış.
- FAILED : Hata almış. Hata açıklaması ERROR_TEXT kolonundan görüntülenir.

MyBatis

Standart Tablo Kolonları

BSA altyapısını kullanan tüm uygulamaların tabloları aşağıdaki kolonları olmak zorundadır.

```
ID VARCHAR2(36 BYTE) NOT NULL,  
STATUS CHAR(1 BYTE) NOT NULL,  
REQUEST_ID VARCHAR2(50 BYTE) NOT NULL,  
CREATE_DATE TIMESTAMP(9) NOT NULL,  
MODIFIED_DATE TIMESTAMP(9) NOT NULL,  
USER_ID VARCHAR2(50 BYTE) NOT NULL,  
TERMINAL VARCHAR2(50 BYTE) NOT NULL,  
ENTITY_ID VARCHAR2(50 BYTE) NOT NULL
```

- "ID" kolu her tablo için primary key olmalıdır.
- Database kayıtları için hiç bir zaman fiziksel delete yapılmaz. "STATUS" kolonu "0" olarak güncellenir. Kayıtları select ederken koşul olarak her zaman "STATUS" = 1 kontrolü eklenmelidir.
- "MODIFIED_DATE" alanı kayıt insert edildiğinde veya değiştiğinde o anki zaman ile güncellenir. BSA herhangi bir kayıt için update işlemi yapıldığında eğer update edilecek kayıt bulunamazsa "NoDataFoundException"(*"Kayıt başka bir kullanıcı tarafından silinmiştir."*) fırlatır. Optimistic locking yapabilmek için update query'sine "MODIFIED_DATE" = <current_time> koşulu eklenmesi yeterli olacaktır. Böylece kayıt başka biri tarafından daha önceden güncellenmiş veya silinmişse kayıt bulunamayacağından update işlemi yapılamayacak ve son kullanıcı bilgilendirilmiş olacaktır. <current_time> değeri *DateUtil.getCurrentTimeInNanos()*

MyBatis Generator Kullanımı

Öncelikle MBG'nin database'e bağlanabilmesi için Eclipse üzerinde Window\Preferences\WhitePlugin Preferences\Ibatis ekranında tablonun owner'ı olan veritabanı kullanıcısının bilgileri girilmelidir. Uygulama sunucusuna tanımlanan datasource'un kullanıcısı tablo owner'ından farklı bir kullanıcıdır. Uygulama kullanıcısı ile tablo owner'ı olan database kullanıcı kavramları birbirine karıştırılmamalıdır. Uygulama kullanıcısının runtime'da tablolara erişebilmesi için synonym yaratılmalı ve owner schema'dan grant verilmelidir. Örnek script'ler aşağıdaki gibidir:

```
create or replace SYNONYM IAM_TEMPLOYEE FOR TIAM.IAM_TEMPLOYEE; --Uygulama kullanıcısı ile çalıştırılmalı.  
  
GRANT DELETE, INSERT, SELECT, UPDATE ON IAM_TEMPLOYEE TO TBSAUSR1; --Owner kullanıcı ile çalıştırılmalı.
```

Doğru database driver'larının kullanılabilmesi için Database type doğru database tipi seçilmelidir.

Örnek bağlantı bilgileri:

Connection URL: jdbc:oracle:thin:@(DESCRIPTION=(LOAD_BALANCE=on)(ADDRESS=(PROTOCOL=TCP)(HOST=tstorat-scan.akbank.com.tr)(PORT=1525))(ADDRESS=(PROTOCOL=TCP)(HOST=tstoratdg11.akbank.com.tr)(PORT=1525))(CONNECT_DATA=(SERVICE_NAME=TTST01_BSA)))

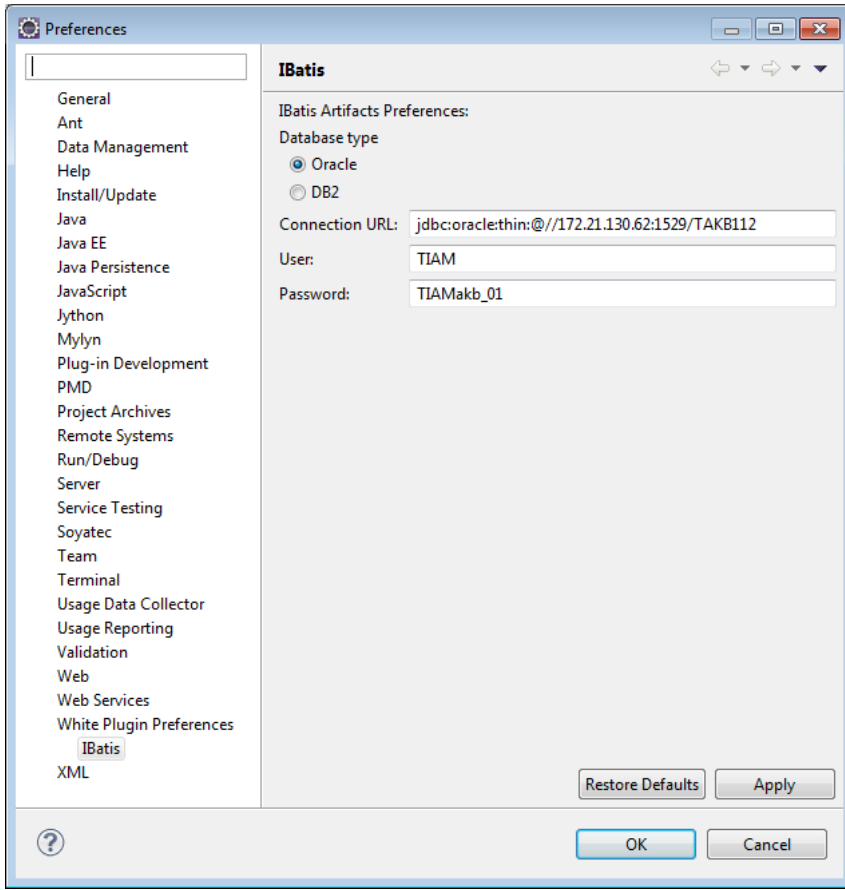
User: IBSA

Password: bbsaa_13

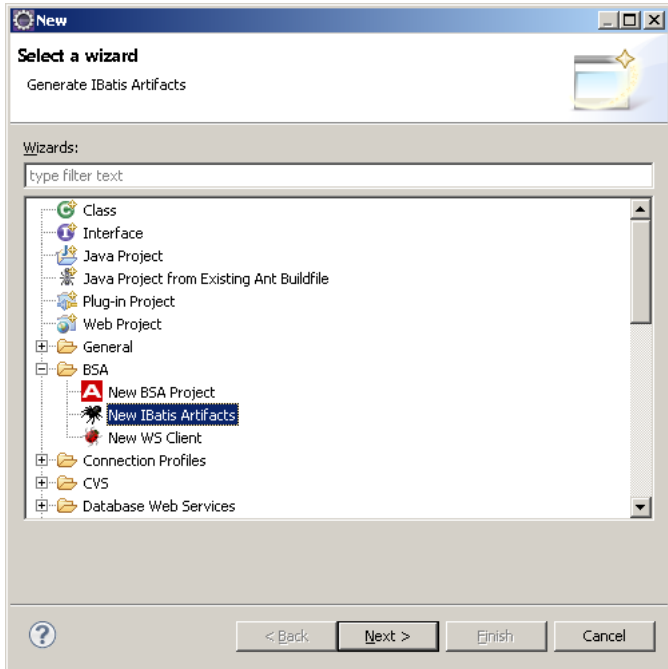
Connection URL: jdbc:db2://172.21.124.72:60000/DB2DB3TS

User: A525208

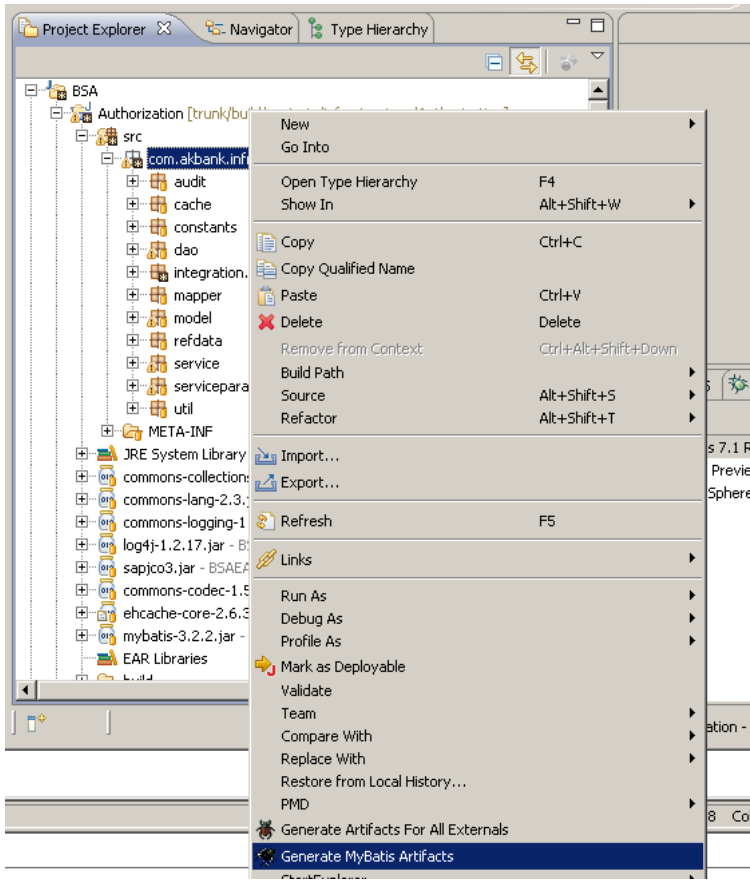
Password: UTK57DEV



MBG'yi çalıştırmak için Eclipse\File\New\Other\BSA\New IBatis Artifacts altından wizard açılabilir.



Veya Project Explorer View'inde herhangi bir package üstünde sağ tıkladıktan sonra çıkan menüde "Generate MyBatis Artifacts" ile açılabilir.



Açılan pencerede Table Name ya da Custom Select Sql'in sadece biri için kod generate edilebilir. Table Name ya da Custom Select Sql'den sadece birinin girilmesi gereklidir.

Table Name girilip generate edilirse o tablo için tüm metotlar mapper.xml ve mapper.java dosyalarında generate edilir.

Custom Select Sql girilip generate edilirse mapper.xml ve mapper.java dosyalarında sadece bu select statement için metot generate edilir.

Custom Select Sql için generate edilecekse Method Name de girilmelidir.

MBG seçilen package altına otomatik olarak eğer yoksa model ve mapper adında iki package yaratacaktır. Servis ve dao class'ları generate edilirse ve eğer yoksa service, serviceparam ve dao package'ları da otomatik oluşturulacaktır.

Model nesneleri model package altına, Mapper.xml ve Mapper interface dosyaları mapper package'ı altına yaratılacaktır. Bu dosyalar generate edildiği için modifiye edilmemelidir. Servis class'ı service package'ı altında, dao class'ı dao package'ı altında, listWrapper class'ı serviceparam package'ı altında oluşturulacaktır.

Subpackage girilirse;

Mapper class'ları \${rootpackage}.mapper.\${subpackage} altında

Model class'ları \${rootpackage}.model.\${subpackage} altında

Service class'ları \${rootpackage}.service.\${subpackage} altında

Dao class'ları \${rootpackage}.dao.\${subpackage} altında

Serviceparam class'ı \${rootpackage}.serviceparam.\${subpackage} altında generate edilir.

Table Name girilecekse şema adı olmadan tablo adı yazılmalıdır.

Custom Select Sql girilecekse select sql şu şekilde olmalıdır.

Select sql where koşullarındaki değişkenler dışında geçerli bir select cümlecisi olmalıdır. Örneğin, aşağıdaki sql için kod generate edileceğini farzedelim.

```
select t1.column1, t2.column2<br>from table1 t1 inner join table2 t2<br>on t1.column3 = t2.column3<br>where t1.status = '1'<br>and t2.status = '1'<br>and t1.column4 = 'ABC'<br>and t2.
```

Farzedelim ki, t1.status, t2.status, t1.column4 ve t2.column5 sql cümlecisine parametre olarak geçirilecek olsun ve t1.column6 sql cümlecisinde hardcoded şekilde 'GHI' olarak kalacak olsun.

Sql cümlecisine t1.status ve t2.status için ortak bir status parametresi geçireceğimizi farzedelim.

Eğer t1.status ve t2.status için farklı status bilgileri geçirilmek isteniyorsa status1 ve status2 gibi ayrı isimler verilebilir. (... where t1.status = #{status1} and t2.status = #{status2} ...)

Bu durumda, Custom Select Sql alanına şu select cümlecği yazılmalıdır.

```
select t1.column1, t2.column2  
from table1 t1 inner join table2 t2  
on t1.column3 = t2.column3  
where t1.status = #{status}<br>and t2.status = #{status}<br>and t1.column4 = #{co
```

Bu durum için, Input Model Class'ında adları status, col4 ve col5 olan 3 tane field generate edilecektir.

Eğer, status'u status1 ve status2 gibi iki farklı isimle verirsek, bu durumda, Input Model Class'ında adları status1, status2, col4 ve col5 olan 4 tane field generate edilecektir.

Eğer Custom Select Sql için kod generate edilecekse Method Name girilmelidir. Bu mapper.java'daki metodun ve mapper.xml'deki select cümlecğinin id'si olacaktır.

Domain Object Name, select sorgusunun resultMap'ının map edileceği Java model class'ının adı olacaktır.

Root Interface uygulamanın kullanacağı datasource'u belirtir. Uygulama datasource'ları BSA'da tanımlanır.

Next butonu ile "Column Override and Generate Input Model" adımına geçilir. Bu ekranda tablo kolonlarının karşılığı olan model nesnesinde oluşacak field'ların tipleri override edilebilir. Default'da mybatis'in JDBC tiplerine karşılık gelen java tipleri kullanılır. Custom Select Sql için kod generate ediliyorsa ve select sql'e en az 1 parametre geçirilecekse Input Model Class'ında oluşturulacak field'ların Jdbc tipleri, Java tipleri ve çoklu değer olarak kullanılıp kullanılmayacağı belirtilir.

Ekran görüntüsünde görüldüğü gibi "APP_STATUS" kolonu (VARCHAR) tipindedir. Java Type default olarak bırakılırsa model nesnesinde String tipinde bir field oluşturulur. boolean olarak seçildiğinde model nesnesinde boolean tipinde bir field oluşturulur. Boolean true olduğunda database'de bu alan "1", false olduğunda "0" olarak tutulur. Select Input Fields kısmındaki field'lar bir önceki sayfada girilen custom select sql'de #{ } içinde verilmiş olan field'lardır. Bu field'lar sql cümlecğine parametre olarak geçirilecektir.

New iBatis Artifacts Wizard

Override Columns and Generate Input Model

Select columns to be overridden and Select input fields to be generated

Select columns

Column Name	Data Type	Java Type
IS_MONETARY	CHAR	Default
IS_APPROVAL_NECESSARY	CHAR	Default
RESOURCE_STATUS	CHAR	Default
PARENT_CODE	VARCHAR	Default
ORDER_NO	NUMERIC	Default
LINK	VARCHAR	Default
TARGET	VARCHAR	Default
IS_ADMINISTRATIVE	CHAR	Default
APP_EXTENSION	CLOB	Default

Select input fields

Input Field Name	Jdbc Type	Java Type	Multi Valued?	Collection Type
code	VARCHAR	String	<input type="checkbox"/>	
name	VARCHAR	String	<input type="checkbox"/>	
resourceStatus	CHAR	boolean	<input type="checkbox"/>	
orderNo	INTEGER	BigDecimal	<input checked="" type="checkbox"/>	List

? < Back Next > Finish Cancel

Next butonu ile "Generate Service and Dao Classes" adımına geçilir. Bu ekranda tablo için service ve dao class'ları generate edilebilir. Dao class'ında hangi metotların (select, insert, update, delete) generate edileceği seçilir. Service Class Name seçili ise dao class'ında generate edilecek metotlar için service class'ında Inbound servisler generate edilir. Service Class Name seçili değilse service class'ı generate edilmez. Dao Class Name seçili değilse dao ve service class'ları generate edilmez. Service class'ındaki select servisi response olarak model nesnesi listesi döner. Bu nedenle bu metot için model nesnesi listesinin wrapper class'ı generate edilir. Generate edilmek istenilen service, dao veya listwrapper class'larıyla aynı adlı class'lar mevcutta varsa bunların overwrite edileceği uyarısı yapılır ve istenilirse işlem iptal edilebilir. "Column Override and Generate Input Model" sayfasına geri dönülüp Finish denilirse dao ve service class'ları generate edilmez.

Custom Select Sql için kod generate ediliyorsa sadece Select metodu seçilebilir.

New iBatis Artifacts Wizard

Generate Service And Dao Classes

Please enter service and dao class names

☒ Service Class Name DemoEmployeeService

☒ Dao Class Name DemoEmployeeDao

Methods to be generated: ☒ Select

☒ Insert

☒ Update

☒ Delete

? < Back Next > Finish Cancel

Hem tablo için hem de custom select sql için kod generate edildiğinde select query'si ile birlikte paging yapılabilecek select...WithPagination adında yeni bir query de generate edilecektir. Bu query kullanılmak istenirse bu query'ye offset ve limit bilgileri parametre olarak verilmelidir. Select edilecek kayıtlar (row number >= offset && row number < offset + limit) olarak select edilecektir. Pagination için select...WithPagination query'sine order by verilmesi önerilmektedir. (Tablo için generate edildiğinde example objesinde orderBy set edilmeli. Custom sql için generate edilirken custom sql'de order by olmalı.)

Üretilen MyBatis Kodlarının Örnek Kullanımı

Metotlardaki mapper nesnesine aşağıdaki kod satırı ile erişilebilir.

```
private static ResourceMapper mapper = MapperFactory.getMapper(ResourceMapper.class);
```


- Insert örneği:

```
public static void insert(Resource record) {
    mapper.insert(record);
}
```

- Update örneği:

```
public static void update(String id,Resource record) {

    ResourceExample example = new ResourceExample();
    Criteria criteria = example.createCriteria();
    criteria.andIdEqualTo(id);
    mapper.updateByExampleSelective(record, example );
}
```

- Liste sorgulama örneği:

```
public static List<Resource> retrieveResources(Resource resource) {

    ResourceExample example = new ResourceExample();
    Criteria c = example.createCriteria();
    c.andStatusEqualTo("1");
    if (resource.getAppId()!=null)
        c.andAppIdEqualTo(resource.getAppId());
    if (StringUtils.isEmpty(resource.getCode())) {
        c.andCodeEqualTo(resource.getCode());
    }
    if (StringUtils.isEmpty(resource.getName())) {
        c.andNameEqualTo(resource.getName());
    }
    if (StringUtils.isEmpty(resource.getType())) {
        c.andTypeEqualTo(resource.getType());
    }
    return mapper.selectByExample(example);
}
```

- Delete örneği:

```
public static void delete(String id) {

    Resource record = new Resource();
    record.setId(id);
    record.setStatus("0");
    mapper.updateByPrimaryKeySelective(record);
}
```

- Tek kayıt sorgulama örneği:

```
public static Resource retrieveResource(String resourceId) {

    ResourceExample example = new ResourceExample();
    Criteria c = example.createCriteria();
    c.andIdEqualTo(resourceId);
    c.andStatusEqualTo("1");
    List<Resource> list = mapper.selectByExampleWithBLOBs(example);
    if (!list.isEmpty()) {
        return list.get(0);
    }
    return null;
}
```

Manuel MyBatis Sorguları Hazırlamak

- mapper package'ı altına <tablo_adi>MapperQuery adında bir interface oluşturulur. Kullanılmak istenen datasource'a ait interface extend edilir. Örnek:

```
public interface EmployeeTransferMapperQuery extends OracleInterface {

    List<BranchApplicationUser> selectBranchApplicationUsers(@Param("registrationNumber") String registrationNumber, @Param("districtCode") String districtCode,@Param("exists") boolean exists);

    public List<EmployeeTransferPrm> selectEmployeeTransfers(@Param("registrationNumber") String registrationNumber, @Param("startDate") Date startDate,
        @Param("endDate") Date endDate);
}
```

- Yine mapper package'ı altına <tablo_adi>MapperQuery.xml adında bir xml oluşturulur. Örnek:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org/DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.akbank.infrastructure.authorization.mapper.EmployeeTransferMapperQuery">
    <resultMap id="BaseResultMap" type="com.akbank.infrastructure.authorization.serviceparam.EmployeeTransferPrm">
        <result column="REGISTRATION_NUMBER" jdbcType="VARCHAR" property="registrationNumber" />
        <result column="FIRST_NAME" jdbcType="VARCHAR" property="firstName" />
        <result column="LAST_NAME" jdbcType="VARCHAR" property="lastName" />
        <result column="OLD_BRANCH_CODE" jdbcType="VARCHAR" property="oldBranchCode" />
        <result column="NEW_BRANCH_CODE" jdbcType="VARCHAR" property="newBranchCode" />
        <result column="OLD_BRANCH_SATELLITE_NUMBER" jdbcType="VARCHAR" property="oldBranchSatelliteNumber" />
        <result column="NEW_BRANCH_SATELLITE_NUMBER" jdbcType="VARCHAR" property="newBranchSatelliteNumber" />
        <result column="START_DATE" jdbcType="TIMESTAMP" property="startDate" />
        <result column="END_DATE" jdbcType="TIMESTAMP" property="endDate" />
    </resultMap>

    <resultMap id="branchApplicationUserMap" type="com.akbank.infrastructure.authorization.model.BranchApplicationUser">
        <result column="REGISTRATION_NUMBER" jdbcType="VARCHAR" property="registrationNumber" />
        <result column="FIRST_NAME" jdbcType="VARCHAR" property="firstName" />
        <result column="LAST_NAME" jdbcType="VARCHAR" property="lastName" />
        <result column="UNIT_CODE" jdbcType="VARCHAR" property="branchCode" />
        <result column="USER_ID" jdbcType="VARCHAR" property="userId" />
        <result column="APP_EXTENSION" jdbcType="VARCHAR" property="appExtension" />
    </resultMap>
</mapper>
```

```

</resultMap>

<select id="selectEmployeeTransfers" parameterType="map" resultMap="BaseResultMap">
    SELECT E.REGISTRATION_NUMBER, E.FIRST_NAME, E.LAST_NAME, E.UNIT_CODE OLD_BRANCH_CODE, ET.NEW_BRANCH_CODE,
           ET.OLD_BRANCH_SATELLITE_NUMBER, ET.NEW_BRANCH_SATELLITE_NUMBER, ET.START_DATE, ET.END_DATE
    FROM IAM_TEMPLOYEE E, IAM_EMPLOYEE_TRANSFER ET
    WHERE E.REGISTRATION_NUMBER = ET.REGISTRATION_NUMBER
           AND E.STATUS='1'
           AND ET.STATUS='1'

    <if test="registrationNumber != null">
        AND E.REGISTRATION_NUMBER = #{registrationNumber,jdbcType=VARCHAR}
    </if>

    <if test="startDate != null">
        AND ET.END_DATE >= #{startDate,jdbcType=TIMESTAMP}
    </if>

    <if test="endDate != null">
        AND ET.START_DATE <![CDATA[<=]]> #{endDate,jdbcType=TIMESTAMP}
    </if>
</select>

<select id="selectBranchApplicationUsers" resultMap="branchApplicationUserMap">
    SELECT E.REGISTRATION_NUMBER, E.FIRST_NAME, E.LAST_NAME, E.UNIT_CODE, U.ID USER_ID, U.APP_EXTENSION
    FROM IAM_TEMPLOYEE E, IAM_TUSER U, IAM_TAPPLICATION A
    WHERE E.REGISTRATION_NUMBER = U.REGISTRATION_NUMBER
           AND U.APP_ID = A.ID
           AND E.STATUS = '1'
           AND U.STATUS = '1'
           AND U.USER_STATUS = '1'
           AND A.CODE = 'INTEGRO'
    <if test="registrationNumber != null">
        AND E.REGISTRATION_NUMBER = #{registrationNumber,jdbcType=VARCHAR}
    </if>

    <if test="districtCode != null">
        AND E.PARENT_UNIT_CODE = #{districtCode,jdbcType=VARCHAR}
    </if>

    <choose>
        <when test="exists==true">
            AND EXISTS (SELECT NULL FROM IAM_EMPLOYEE_TRANSFER ET WHERE ET.REGISTRATION_NUMBER=E.REGISTRATION_NUMBER)
        </when>
        <otherwise>
            AND NOT EXISTS (SELECT NULL FROM IAM_EMPLOYEE_TRANSFER ET WHERE ET.REGISTRATION_NUMBER=E.REGISTRATION_NUMBER)
        </otherwise>
    </choose>
</select>
</mapper>

```

■ Karmaşık resultMap kullanımı

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.akbank.infrastructure.authorization.mapper.UserMapperQuery">

    <resultMap id="BaseResultMap" type="com.akbank.infrastructure.authorization.cache.model.UserRight">
        <result column="APP_USERNAME" jdbcType="VARCHAR" property="appUsername" />
        <result column="APP_ID" jdbcType="VARCHAR" property="appId" />
        <result column="REGISTRATION_NUMBER" jdbcType="VARCHAR" property="registrationNumber" />
        <collection property="rightList" ofType="com.akbank.infrastructure.authorization.cache.model.RightItem" resultMap="rightResultMap" columnPrefix="right_" />
    </resultMap>

    <resultMap id="rightResultMap" type="com.akbank.infrastructure.authorization.cache.model.RightItem">
        <result column="CODE" jdbcType="VARCHAR" property="code" />
        <result column="DESCRIPTION" jdbcType="VARCHAR" property="description" />
        <result column="TYPE" jdbcType="VARCHAR" property="type" />
        <result column="RIGHT_TYPE" jdbcType="VARCHAR" property="rightType" />
        <result column="IS_MONETARY" jdbcType="CHAR" property="isMonetary" typeHandler="com.akbank.bsa.mybatis.typehandler.StringBooleanTypeHandler" />
        <result column="IS_APPROVAL_NECESSARY" jdbcType="CHAR" property="isApprovalNecessary" typeHandler="com.akbank.bsa.mybatis.typehandler.StringBooleanTypeHandler" />
    </resultMap>

    <select id="retrieveUsersWithRolesAndResources" resultMap="BaseResultMap">
        select * from
            (select
                'role' AS right_type,
                roles_view.APP_ID,
                roles_view.APP_USERNAME,
                roles_view.REGISTRATION_NUMBER,
                roles_view.ROLE_CODE AS RIGHT_CODE,
                roles_view.ROLE_DESCRIPTION AS RIGHT_DESCRIPTION,
                'x' as right_right_TYPE,
                'x' as right_IS_MONETARY,
                'x' as right_IS_APPROVAL_NECESSARY
            from IAM_VUSER_ROLES roles_view
            union all
            select
                'resource' AS right_type,
                resources_view.APP_ID,
                resources_view.APP_USERNAME,
                resources_view.REGISTRATION_NUMBER,
                resources_view.RESOURCE_CODE AS RIGHT_CODE,
                resources_view.RESOURCE_DESCRIPTION AS RIGHT_DESCRIPTION,
                RE.TYPE as right_right_TYPE,
                RE.IS_MONETARY as right_IS_MONETARY,
                RE.IS_APPROVAL_NECESSARY as right_IS_APPROVAL_NECESSARY
            from IAM_VUSER_RESOURCES resources_view, IAM_TRESOURCE re
            where RE.STATUS = '1' and RE.ID = resources_view.RESOURCE_ID)
        where 1=1

        <if test="appId != null">
            AND APP_ID = #{appId,jdbcType=VARCHAR}
        </if>
        <if test="appUsername != null">
            AND APP_USERNAME = #{appUsername,jdbcType=VARCHAR}
        </if>
    </select>

```

- Daha ayrıntılı bilgi edinmek için: <http://mybatis.github.io/mybatis-3/sqlmap-xml.html>

Database'deki bir kayıdın üstünde yapılan değişikliklerin diğer bir tabloda saklanmasını sağlar. Bu özelliği kullanabilmek için:

- Eclipse'de "Generate MyBatis Artifacts" wizard'ı açılır.
- Açılan pencerede "Enable auto history features" checkbox'ı check'lenir ve kayıt üzerinde yapılan değişikliklerin saklanacağı tablo adı yazılır.

- Generate edilen kodlar aşağıdaki ekran görüntüsündeki gibi oluşur:

```
@Historical(sourceTable="IAM_TRESOURCE", destinationTable="H_IAM_TRESOURCE")
int updateByExampleSelective(@Param("record") Resource record, @Param("example") ResourceExample example);

/**
 * This method was generated by MyBatis Generator.
 * This method corresponds to the database table IAM_TRESOURCE
 *
 * @mbggenerated Wed Jul 03 10:40:14 EEST 2013
 */
@Historical(sourceTable="IAM_TRESOURCE", destinationTable="H_IAM_TRESOURCE")
int updateByExampleWithBLOBs(@Param("record") Resource record, @Param("example") ResourceExample example);

/**
 * This method was generated by MyBatis Generator.
 * This method corresponds to the database table IAM_TRESOURCE
 *
 * @mbggenerated Wed Jul 03 10:40:14 EEST 2013
 */
@Historical(sourceTable="IAM_TRESOURCE", destinationTable="H_IAM_TRESOURCE")
int updateByExample(@Param("record") Resource record, @Param("example") ResourceExample example);
```

- History tablosu ana kayıtların tutulduğu tablonun bire bir aynı yapıda tablosu olmalıdır. Sadece History tablosuna 2 ek kolon eklenir (**EXPIRED_DATE**, **OPERATION_TYPE**). Örnek sql script aşağıdaki gibidir:
- History tablosunun adının başında "H_" olmalıdır. (tablo isimlendirme standardı)

```
CREATE TABLE IAM_TRESOURCE
(
  ID VARCHAR2(36 BYTE) NOT NULL,
  STATUS CHAR(1 BYTE) NOT NULL,
  REQUEST_ID VARCHAR2(50 BYTE) NOT NULL,
  CREATE_DATE TIMESTAMP(9) NOT NULL,
  MODIFIED_DATE TIMESTAMP(9) NOT NULL,
  USER_ID VARCHAR2(50 BYTE) NOT NULL,
  TERMINAL VARCHAR2(50 BYTE) NOT NULL,
  ENTITY_ID VARCHAR2(50 BYTE) NOT NULL,
  APP_ID VARCHAR2(36 BYTE) NOT NULL,
  CODE VARCHAR2(50 BYTE) NOT NULL,
  NAME VARCHAR2(100 BYTE),
  DESCRIPTION VARCHAR2(250 BYTE),
  TYPE VARCHAR2(50 BYTE) NOT NULL,
  REFERENCE_NUMBER VARCHAR2(50 BYTE),
  IS_MONETARY CHAR(1 BYTE) NOT NULL,
  IS_APPROVAL_NECESSARY CHAR(1 BYTE) NOT NULL,
  APP_EXTENSION CLOB
);

CREATE TABLE H_IAM_TRESOURCE
(
  ID VARCHAR2(36 BYTE) NOT NULL,
  STATUS CHAR(1 BYTE) NOT NULL,
  REQUEST_ID VARCHAR2(50 BYTE),
  CREATE_DATE TIMESTAMP(9),
  MODIFIED_DATE TIMESTAMP(9),
  USER_ID VARCHAR2(50 BYTE),
  TERMINAL VARCHAR2(50 BYTE),
  ENTITY_ID VARCHAR2(50 BYTE),
  APP_ID VARCHAR2(36 BYTE) NOT NULL,
  CODE VARCHAR2(50 BYTE) NOT NULL,
```

```
NAME VARCHAR2(100 BYTE),
DESCRIPTION VARCHAR2(250 BYTE),
TYPE VARCHAR2(50 BYTE),
REFERENCE_NUMBER VARCHAR2(50 BYTE),
IS_MONETARY CHAR(1 BYTE) NOT NULL,
IS_APPROVAL_NECESSARY CHAR(1 BYTE) NOT NULL,
APP_EXTENSION CLOB,
EXPIRED_DATE TIMESTAMP(9),
OPERATION_TYPE VARCHAR2(12 BYTE)
);
```

- Generate edilen kodlarda @Historical annotation'ı otomatik olarak oluşturulur. Ancak manuel yazılan sqlmap xml dosyalarında eğer insert, update ve delete metodlarının başına @Historical annotation'ı manuel olarak eklenmelidir.

MQ

Yeni Queue ve Queue Connection Tanımı

BSA; Websphere MQ Server ile mesajlaşma kabiliyetine sahiptir.Queue Manager tanımı ve Queue tanımları aşağıdaki tablolarda yer almaktadır.

- BSA_QUEUE_CONNECTION_DEF

NAME	JNDI	PORT	HOSTNAME	CHANNEL	QUEUE_MANAGER	CCSID	TRANSPORT_TYPE	USERNAME	PASSWORD	CLASS_NAME
BSAQCF	jms/bsaQueueCF	1421	172.21.131.36	BSACHANNEL	MQAPPST	819	1	bsausr	1	com.akbank.bsa.core.jms.cf.BSAQueueManager

- BSA_QUEUE_DEFINITION

ALIAS	NAME	QUEUE_CONNECTION_NAME	JNDI_NAME	IS_ENABLED
ASYNC_QUEUE	SIT_ASYNC_QUEUE	BSAQCF	jms/sitAsyncQueue	1
BACKOUT_QUEUE	SIT_BACKOUT_QUEUE	BSAQCF	jms/sitBackoutQueue	1

- Yeni bir Queue Connection tanımı yapmak için Queue Connection bilgilerinin BSA ekibine iletilmesi gerekmektedir.
- Queue tanımları BSA_QUEUE_DEFINITION tablosuna girilmelidir. Hangi Queue Connection'ını kullanacak ise QUEUE_CONNECTION_NAME kolonuna o yazılmalıdır.
- SIT, UAT, PROD geçişlerinde yeni tanımlanan Queue ve Queue Connection'lar için Websphere'de tanımlar yapılmalıdır. Geçiş öncesinde bu tanımlar da BSA ekibine iletilmelidir.

@MqConsumer

Mesaj dinlemek için BSA servis metodlarının başına @MQConsumer annotation'ını eklenmelidir.

MQ Message'a inBag'deki "MQ_MESSAGE" key'i ile erişilir.

```
@Service(value = "DEMO_DEMO_PROJECT_ONE_QUEUE")
@MqConsumer(inboundQueue="AKARR.UPD.RQ")
public static Bag createCustomer(Bag inBag) throws BSAException {
    String mqMessage = inBag.get("MQ_MESSAGE").toString();
    log.info(mqMessage);
    return null;
}
```

@MqConsumer Attributes

- String inboundQueue(); **Dinlenecek olan Queue tanımının adıdır.**
- boolean sessionTransacted() default false; **Transactional get özelliğidir.**
- int maxConcurrentConsumers() default 3; **Queue'yu dinleyen thread sayısıdır.**

Mesaj Göndermek

Mesaj göndermek için MqFactory class'ı kullanılır.

Mesaj gönderebilmek için öncelikle QUEUE_CONNECTIN_DEFINITION tablosundaki class_name kolonundaki class'ı extend eden bir class yaratılmalıdır.

```
package com.akbank.demo.demoprojectone.mq;

import com.akbank.bsa.core.jms.cf.BSAQueueManager;

public class DemoMqManager extends BSAQueueManager {
}
```

Örnek kod:

```
@Service(value = "DEMO_DEMO_PROJECT_ONE_SEND_MESSAGE")
@Inbound
public static void sendMessage(String message) throws BSAException {
    DemoMqManager producer = MqFactory.getProducer(DemoMqManager.class);
    producer.sendMessage(message, "AKARR.UPD.RQ");
}
```

Web Servis