```
transformation SpringToExtendedSpring(in springModel : SpringModel, out extendedSpringModel : ExtendedSpringModel);

modeltype SpringModel "strict" uses springConfigDsl('http://www.xtext.org/spring/SpringConfigDsl');
modeltype ExtendedSpringModel "strict" uses javaComponents('http://www.ema.fr/beans/metamodel/javaComponents');




////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
//                                MAIN                                        //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////


main() {
    springModel.rootObjects() [SpringModel::Configuration]->map toExtendedModel();
}



////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
//                                MAPPINGS                                    //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////


/////////////////////////////////////////////////////////////////
//                      Configuration                          //
/////////////////////////////////////////////////////////////////

mapping SpringModel::Configuration::toExtendedModel() : ExtendedSpringModel::springConfigDsl::Configuration {
    defaultAutowire := self.defaultAutowire.getDefaultAutoWire();
    defaultInitMethod := self.defaultInitMethod;
    defaultAutowireCandidates := self.defaultAutowireCandidates;
    defaultDestroyMethod := self.defaultDestroyMethod;
    defaultLazyInit := self.defaultLazyInit.getDefaultableBoolean();
    defaultMerge := self.defaultMerge.getDefaultableBoolean();
    profile := self.profile;
    description := self.description;
    components := self.components->map toComponent();//getComponents();
    alias := self.alias->map toAlias();//getAliases();
    imports := self.imports->map toImport();//getImports();
    contexts := self.contexts->map toContext();//getContexts();
    aspects := self.aspects->map toAspect();//getAspects();
    utilConstants := self.utilConstants->map toUtilConstant();//getAbstractArtefacts???
    utilLists := self.utilLists->map toUtilList();//getAbstractArtefacts???
```

```
59      utilMaps := self.utilMaps->map toUtilMap();//getAbstractArtefacts???
60      utilProperties := self.utilProperties->map
    toUtilProperties();//getAbstractArtefacts???
61      utilSets := self.utilSets->map toUtilSet(); //getAbstractArtefacts???
62      utilPropertiesPath := self.utilPropertiesPath->map
    toUtilPropertyPath();//getUtilPropertiesPath();
63      txAdvices := self.txAdvices->map toTxAdvise();//getTxAdvices();
64      txJtaTransactionManager := self.txJtaTransactionManager.map
    toTxJtaTransactionManager();
65      ConfigurationComposite := self.ConfigurationComposite->map
    toExtendedModel();//getConfigurations();
66  }
67
68
69  ////////////////////////////////////////////////////////////////
70  //                      Component                             //
71  ////////////////////////////////////////////////////////////////
72
73  mapping SpringModel::Component::toComponent() : ExtendedSpringModel::Comp {
74      name := self.name;
75      names := self.names;
76      _class := self._class.map toCreationMethod();
77      features := self.features->map toFeature();//getFeatures();
78      _abstract := self._abstract.getBoolean();
79      autowireCandidate := self.autowireCandidate.getDefaultableBoolean();
80      autowire := self.autowire.getDefaultableBoolean();
81      dependsOn := self.dependsOn.map toComponent();
82      detroyMethod := self.detroyMethod;
83      initMethod := self.initMethod;
84      lazyInit := self.lazyInit.getDefaultableBoolean();
85      parent := self.parent.map toComponent();
86      primary := self.primary.getBoolean();
87      scope := self.scope;
88      description := self.description;
89      lookupMethods := self.lookupMethods->map toLookupMethod();//getLookupMethods();
90      qualifiers := self.qualifiers->map toQualifier();//getQualifiers();
91      meta := self.meta->map toMeta();//getMetas();
92      aopScopedProxy := self.aopScopedProxy.map toAopScopedProxy();
93      utilPropertiesPath := self.utilPropertiesPath->map
    toUtilPropertyPath();//getUtilPropertiesPath();
94  }
95
96
97  ////////////////////////////////////////////////////////////////
98  //                      CreationMethod                        //
99  ////////////////////////////////////////////////////////////////
100
101 mapping SpringModel::CreationMethod::toCreationMethod() :
    ExtendedSpringModel::springConfigDsl::CreationMethod {
102     factoryMethod := self.factoryMethod;
103     _class := self._class.map toClassOrFactory();
104 }
105
106
107 ////////////////////////////////////////////////////////////////
108 //                      ClassOrFactory                        //
109 ////////////////////////////////////////////////////////////////
110
111 mapping SpringModel::ClassOrFactory::toClassOrFactory() :
    ExtendedSpringModel::springConfigDsl::ClassOrFactory {
112     if(self.oclIsTypeOf(SpringModel::Class))
113     {
```

```
114         self.oclAsType(SpringModel::Class).map toClass();
115     }
116     else
117     {
118         self.oclAsType(SpringModel::Factory).map toFactory();
119     }
120 }
121
122
123 ////////////////////////////////////////////////////////////////
124 //                      Class                                 //
125 ////////////////////////////////////////////////////////////////
126
127 mapping SpringModel::Class::toClass() : ExtendedSpringModel::springConfigDsl::Class {
128     classname := self.classname;
129 }
130
131
132 ////////////////////////////////////////////////////////////////
133 //                      Factory                               //
134 ////////////////////////////////////////////////////////////////
135
136 mapping SpringModel::Factory::toFactory() : ExtendedSpringModel::springConfigDsl::Factory
    {
137     factoryBean := self.factoryBean.map toComponent();
138 }
139
140
141 ////////////////////////////////////////////////////////////////
142 //                      Feature                               //
143 ////////////////////////////////////////////////////////////////
144
145 mapping SpringModel::Feature::toFeature() : ExtendedSpringModel::springConfigDsl::Feature
    {
146     name := self.name;
147     artefact := self.artefact.map toAbstractArtefact();
148     description := self.description;
149     index := self.index;
150     type := self.type;
151 }
152
153
154 ////////////////////////////////////////////////////////////////
155 //                      AbstractArtifact                      //
156 ////////////////////////////////////////////////////////////////
157
158 mapping SpringModel::AbstractArtefact::toAbstractArtefact() :
    ExtendedSpringModel::springConfigDsl::AbstractArtefact {
159     if(self.oclIsKindOf(SpringModel::Component))
160     {
161         self.oclAsType(SpringModel::Component).map toComponent();
162     }
163     else if(self.oclIsKindOf(SpringModel::Attribute))
164     {
165         self.oclAsType(SpringModel::Attribute).map toAttribute();
166     }
167     else if(self.oclIsKindOf(SpringModel::Reference))
168     {
169         self.oclAsType(SpringModel::Reference).map toReference();
170     }
171     else
172     {
```

```
173         if(self.oclIsKindOf(SpringModel::sList))
174         {
175             self.oclAsType(SpringModel::sList).map toSList();
176         }
177         else if(self.oclIsKindOf(SpringModel::sSet))
178         {
179             self.oclAsType(SpringModel::sSet).map toSSet();
180         }
181         else if(self.oclIsKindOf(SpringModel::Props))
182         {
183             self.oclAsType(SpringModel::Props).map toProps();
184         }
185         else if(self.oclIsKindOf(SpringModel::Map))
186         {
187             self.oclAsType(SpringModel::Map).map toMap();
188         }
189     }
190 }
191
192
193 ////////////////////////////////////////////////////////////////
194 //                        Attribute                           //
195 ////////////////////////////////////////////////////////////////
196
197 mapping SpringModel::Attribute::toAttribute() :
    ExtendedSpringModel::springConfigDsl::Attribute {
198     if(self.oclIsTypeOf(SpringModel::UtilConstant))
199     {
200         self.oclAsType(SpringModel::UtilConstant).map toUtilConstant();
201     }
202     else
203     {
204         value := self.value;
205         type := self.type;
206     }
207 }
208
209
210 ////////////////////////////////////////////////////////////////
211 //                        UtilConstant                        //
212 ////////////////////////////////////////////////////////////////
213
214 mapping SpringModel::UtilConstant::toUtilConstant() :
    ExtendedSpringModel::springConfigDsl::UtilConstant {
215     value := self.value;
216     type := self.type;
217     StaticField := self.StaticField;
218     name := self.name;
219 }
220
221
222 ////////////////////////////////////////////////////////////////
223 //                        Reference                           //
224 ////////////////////////////////////////////////////////////////
225
226 mapping SpringModel::Reference::toReference() :
    ExtendedSpringModel::springConfigDsl::Reference {
227     ref := self.ref.map toComponent();
228 }
229
230
231 ////////////////////////////////////////////////////////////////
```

```
232 //                      sList ("List" is a reserved word)       //
233 ////////////////////////////////////////////////////////////////////
234
235 mapping SpringModel::sList::toSList() : ExtendedSpringModel::springConfigDsl::sList {
236     if(self.oclIsTypeOf(SpringModel::UtilList))
237     {
238         self.oclAsType(SpringModel::UtilList).map toUtilList();
239     }
240     else
241     {
242         valueType := self.valueType;
243         merge := self.merge.getDefaultableBoolean();
244         description := self.description;
245         artefacts := self.artefacts->map toAbstractArtefact();//getAbstractArtefacts();
246     }
247 }
248
249
250 ////////////////////////////////////////////////////////////////////
251 //                      UtilList                                  //
252 ////////////////////////////////////////////////////////////////////
253
254 mapping SpringModel::UtilList::toUtilList() :
    ExtendedSpringModel::springConfigDsl::UtilList {
255     valueType := self.valueType;
256     merge := self.merge.getDefaultableBoolean();
257     description := self.description;
258     artefacts := self.artefacts->map toAbstractArtefact();//getAbstractArtefacts();
259     name := self.name;
260     listClass := self.listClass;
261     scope := self.scope;
262 }
263
264
265 ////////////////////////////////////////////////////////////////////
266 //                      sSet ("Set" is a reserved word)           //
267 ////////////////////////////////////////////////////////////////////
268
269 mapping SpringModel::sSet::toSSet() : ExtendedSpringModel::springConfigDsl::sSet {
270     if(self.oclIsTypeOf(SpringModel::UtilSet))
271     {
272         self.oclAsType(SpringModel::UtilSet).map toUtilSet();
273     }
274     else
275     {
276         valueType := self.valueType;
277         merge := self.merge.getDefaultableBoolean();
278         description := self.description;
279         artefacts := self.artefacts->map toAbstractArtefact();//getAbstractArtefacts();
280     }
281 }
282
283
284 ////////////////////////////////////////////////////////////////////
285 //                      UtilSet                                   //
286 ////////////////////////////////////////////////////////////////////
287
288 mapping SpringModel::UtilSet::toUtilSet() : ExtendedSpringModel::springConfigDsl::UtilSet
    {
289     valueType := self.valueType;
290     merge := self.merge.getDefaultableBoolean();
291     description := self.description;
```

```
292     artefacts := self.artefacts->map toAbstractArtefact();//getAbstractArtefacts();
293     name := self.name;
294     setClass := self.setClass;
295     scope := self.scope;
296 }
297
298
299 ////////////////////////////////////////////////////////////////
300 //                        Props                               //
301 ////////////////////////////////////////////////////////////////
302
303 mapping SpringModel::Props::toProps() : ExtendedSpringModel::springConfigDsl::Props {
304     if(self.oclIsTypeOf(SpringModel::UtilProperties))
305     {
306         self.oclAsType(SpringModel::UtilProperties).map toUtilProperties();
307     }
308     else
309     {
310         valueType := self.valueType;
311         merge := self.merge.getDefaultableBoolean();
312         description := self.description;
313         props := self.props->map toProp();//getProps();
314     }
315 }
316
317
318 ////////////////////////////////////////////////////////////////
319 //                        UtilProperties                      //
320 ////////////////////////////////////////////////////////////////
321
322 mapping SpringModel::UtilProperties::toUtilProperties() :
    ExtendedSpringModel::springConfigDsl::UtilProperties {
323     valueType := self.valueType;
324     merge := self.merge.getDefaultableBoolean();
325     description := self.description;
326     props := self.props->map toProp();//getProps();
327     propertyfile := self.propertyfile.map toPropertyFile();
328     name := self.name;
329     ignoreResourceNotFound := self.ignoreResourceNotFound.getBoolean();
330     localOverride := self.localOverride.getBoolean();
331     scope := self.scope;
332 }
333
334
335 ////////////////////////////////////////////////////////////////
336 //                        Prop                                //
337 ////////////////////////////////////////////////////////////////
338
339 mapping SpringModel::Prop::toProp() : ExtendedSpringModel::springConfigDsl::Prop {
340     key := self.key;
341     value := self.value;
342 }
343
344
345 ////////////////////////////////////////////////////////////////
346 //                        Map                                 //
347 ////////////////////////////////////////////////////////////////
348
349 mapping SpringModel::Map::toMap() : ExtendedSpringModel::springConfigDsl::Map {
350     valueType := self.valueType;
351     merge := self.merge.getDefaultableBoolean();
352     description := self.description;
```

```
353     keyType := self.keyType;
354     entries := self.entries->map toMapEntry();//getMapEntries();
355 }
356
357
358 ////////////////////////////////////////////////////////////////
359 //                        UtilMap                             //
360 ////////////////////////////////////////////////////////////////
361
362 mapping SpringModel::UtilMap::toUtilMap() : ExtendedSpringModel::springConfigDsl::UtilMap
    {
363     if(self.oclIsTypeOf(SpringModel::UtilMap))
364     {
365         self.oclAsType(SpringModel::UtilMap).map toUtilMap();
366     }
367     else
368     {
369         valueType := self.valueType;
370         merge := self.merge.getDefaultableBoolean();
371         description := self.description;
372         keyType := self.keyType;
373         entries := self.entries->map toMapEntry();//getMapEntries();
374         name := self.name;
375         mapClass := self.mapClass;
376         scope := self.scope;
377     }
378 }
379
380
381 ////////////////////////////////////////////////////////////////
382 //                        MapEntry                            //
383 ////////////////////////////////////////////////////////////////
384
385 mapping SpringModel::MapEntry::toMapEntry() :
    ExtendedSpringModel::springConfigDsl::MapEntry {
386     key := self.key.map toKey();
387     value := self.value.map toAbstractArtefact();
388     description := self.description;
389     valueType := self.valueType;
390 }
391
392
393 ////////////////////////////////////////////////////////////////
394 //                        Key                                 //
395 ////////////////////////////////////////////////////////////////
396
397 mapping SpringModel::Key::toKey() : ExtendedSpringModel::springConfigDsl::Key {
398     description := self.description;
399     key := self.key.map toAbstractKeyValue();
400 }
401
402
403 ////////////////////////////////////////////////////////////////
404 //                        AbstractKeyValue                    //
405 ////////////////////////////////////////////////////////////////
406
407 mapping SpringModel::AbstractKeyValue::toAbstractKeyValue() :
    ExtendedSpringModel::springConfigDsl::AbstractKeyValue {
408     if(self.oclIsKindOf(SpringModel::AbstractArtefact))
409     {
410         self.oclAsType(SpringModel::AbstractArtefact).map toAbstractArtefact();
411     }
```

```
412     else
413     {
414         self.oclAsType(SpringModel::DataString).map toDataString();
415     }
416 }
417
418
419 ////////////////////////////////////////////////////////////////////
420 //                        DataString                              //
421 ////////////////////////////////////////////////////////////////////
422
423 mapping SpringModel::DataString::toDataString() :
    ExtendedSpringModel::springConfigDsl::DataString {
424     s := self.s;
425 }
426
427
428 ////////////////////////////////////////////////////////////////////
429 //                        AopScopedProxy                          //
430 ////////////////////////////////////////////////////////////////////
431
432 mapping SpringModel::AopScopedProxy::toAopScopedProxy() :
    ExtendedSpringModel::springConfigDsl::AopScopedProxy {
433     proxyTargetClass := self.proxyTargetClass;
434 }
435
436
437 ////////////////////////////////////////////////////////////////////
438 //                        LookupMethod                            //
439 ////////////////////////////////////////////////////////////////////
440
441 mapping SpringModel::LookupMethod::toLookupMethod() :
    ExtendedSpringModel::springConfigDsl::LookupMethod {
442     name := self.name;
443     ref := self.ref.map toComponent();
444 }
445
446
447 ////////////////////////////////////////////////////////////////////
448 //                        Qualifier                               //
449 ////////////////////////////////////////////////////////////////////
450
451 mapping SpringModel::Qualifier::toQualifier() :
    ExtendedSpringModel::springConfigDsl::Qualifier {
452     type := self.type;
453     value := self.value;
454     qualifierAttributes := self.qualifierAttributes->map
    toQualifierAttribute();//getQualifierAttributes();
455 }
456
457
458 ////////////////////////////////////////////////////////////////////
459 //                        QualifierAttribute                      //
460 ////////////////////////////////////////////////////////////////////
461
462 mapping SpringModel::QualifierAttribute::toQualifierAttribute() :
    ExtendedSpringModel::springConfigDsl::QualifierAttribute {
463     key := self.key;
464     value := self.value;
465 }
466
467
```

```
468 ////////////////////////////////////////////////////////////////
469 //                        Meta                                //
470 ////////////////////////////////////////////////////////////////
471
472 mapping SpringModel::Meta::toMeta() : ExtendedSpringModel::springConfigDsl::Meta {
473     key := self.key;
474     value := self.value;
475 }
476
477
478 ////////////////////////////////////////////////////////////////
479 //                        UtilPropertyPath                    //
480 ////////////////////////////////////////////////////////////////
481
482 mapping SpringModel::UtilPropertyPath::toUtilPropertyPath() :
    ExtendedSpringModel::springConfigDsl::UtilPropertyPath {
483     name := self.name;
484     path := self.path;
485 }
486
487
488 ////////////////////////////////////////////////////////////////
489 //                        Alias                               //
490 ////////////////////////////////////////////////////////////////
491
492 mapping SpringModel::Alias::toAlias() : ExtendedSpringModel::springConfigDsl::Alias {
493     origin := self.origin.map toComponent();
494     alias := self.alias;
495 }
496
497
498 ////////////////////////////////////////////////////////////////
499 //                        Import                              //
500 ////////////////////////////////////////////////////////////////
501
502 mapping SpringModel::Import::toImport() : ExtendedSpringModel::springConfigDsl::Import {
503     resource := self.resource;
504 }
505
506
507 ////////////////////////////////////////////////////////////////
508 //                        Context                             //
509 ////////////////////////////////////////////////////////////////
510
511 mapping SpringModel::Context::toContext() : ExtendedSpringModel::springConfigDsl::Context
    {
512     if(self.oclIsKindOf(SpringModel::AnnotationConfig))
513     {
514         self.oclAsType(SpringModel::AnnotationConfig).map toAnnotationConfig();
515     }
516     else if(self.oclIsKindOf(SpringModel::ComponentScan))
517     {
518         self.oclAsType(SpringModel::ComponentScan).map toComponentScan();
519     }
520     else if(self.oclIsKindOf(SpringModel::LoadTimeWeaver))
521     {
522         self.oclAsType(SpringModel::LoadTimeWeaver).map toLoadTimeWeaver();
523     }
524     else if(self.oclIsKindOf(SpringModel::MbeanExport))
525     {
526         self.oclAsType(SpringModel::MbeanExport).map toMbeanExport();
527     }
```

```
528     else if(self.oclIsKindOf(SpringModel::MbeanServer))
529     {
530         self.oclAsType(SpringModel::MbeanServer).map toMbeanServer();
531     }
532     else if(self.oclIsKindOf(SpringModel::PropertyHolder))
533     {
534         self.oclAsType(SpringModel::PropertyHolder).map toPropertyHolder();
535     }
536     else if(self.oclIsKindOf(SpringModel::SpringConfigured))
537     {
538         self.oclAsType(SpringModel::SpringConfigured).map toSpringConfigured();
539     }
540 }
541
542
543 ////////////////////////////////////////////////////////////////////
544 //                        AnnotationConfig                        //
545 ////////////////////////////////////////////////////////////////////
546
547 mapping SpringModel::AnnotationConfig::toAnnotationConfig() :
    ExtendedSpringModel::springConfigDsl::AnnotationConfig {
548
549 }
550
551
552 ////////////////////////////////////////////////////////////////////
553 //                        ComponentScan                           //
554 ////////////////////////////////////////////////////////////////////
555
556 mapping SpringModel::ComponentScan::toComponentScan() :
    ExtendedSpringModel::springConfigDsl::ComponentScan {
557     basePackage := self.basePackage;
558     annotationConfig := self.annotationConfig.getBoolean();
559     nameGeneratorBean := self.nameGeneratorBean.map toComponent();
560     resourcePattern := self.resourcePattern;
561     scopeResolver := self.scopeResolver.map toComponent();
562     scopedProxy := self.scopedProxy.getScopedProxy();
563     useDefaultFilters := self.useDefaultFilters.getBoolean();
564     includeFilters := self.includeFilters->map toIncludeFilter();//getIncludeFilters();
565     excludeFilters := self.excludeFilters->map toExcludeFilter();//getExcludeFilters();
566 }
567
568
569 ////////////////////////////////////////////////////////////////////
570 //                        IncludeFilter                           //
571 ////////////////////////////////////////////////////////////////////
572
573 mapping SpringModel::IncludeFilter::toIncludeFilter() :
    ExtendedSpringModel::springConfigDsl::IncludeFilter {
574     type := self.type.getTypeFilter();
575     expression := self.expression;
576 }
577
578
579 ////////////////////////////////////////////////////////////////////
580 //                        ExcludeFilter                           //
581 ////////////////////////////////////////////////////////////////////
582
583 mapping SpringModel::ExcludeFilter::toExcludeFilter() :
    ExtendedSpringModel::springConfigDsl::ExcludeFilter {
584     type := self.type.getTypeFilter();
585     expression := self.expression;
```

```
586 }
587
588
589 ////////////////////////////////////////////////////////////////////
590 //                        LoadTimeWeaver                          //
591 ////////////////////////////////////////////////////////////////////
592
593 mapping SpringModel::LoadTimeWeaver::toLoadTimeWeaver() :
    ExtendedSpringModel::springConfigDsl::LoadTimeWeaver {
594     aspectjWeaving := self.aspectjWeaving;
595     weaverClass := self.weaverClass;
596 }
597
598
599 ////////////////////////////////////////////////////////////////////
600 //                        MbeanExport                             //
601 ////////////////////////////////////////////////////////////////////
602
603 mapping SpringModel::MbeanExport::toMbeanExport() :
    ExtendedSpringModel::springConfigDsl::MbeanExport {
604     defaultDomain := self.defaultDomain;
605     registration := self.registration.getMbeanRegistration();
606     server := self.server.map toComponent();
607 }
608
609
610 ////////////////////////////////////////////////////////////////////
611 //                        MbeanServer                             //
612 ////////////////////////////////////////////////////////////////////
613
614 mapping SpringModel::MbeanServer::toMbeanServer() :
    ExtendedSpringModel::springConfigDsl::MbeanServer {
615     agentId := self.agentId;
616     name := self.name;
617 }
618
619
620 ////////////////////////////////////////////////////////////////////
621 //                        PropertyHolder                          //
622 ////////////////////////////////////////////////////////////////////
623
624 mapping SpringModel::PropertyHolder::toPropertyHolder() :
    ExtendedSpringModel::springConfigDsl::PropertyHolder {
625     if(self.oclIsTypeOf(SpringModel::PropertyPlaceholder))
626     {
627         propertyfile := self.propertyfile.map toPropertyFile();
628         ignoreResourceNotFound := self.ignoreResourceNotFound.getBoolean();
629         ignoreUnresolvable := self.ignoreUnresolvable.getBoolean();
630         localOverride := self.localOverride.getBoolean();
631         order := self.order;
632         propertiesRef := self.propertiesRef.map toComponent();
633     }
634     else
635     {
636         self.oclAsType(SpringModel::PropertyPlaceholder).map toPropertyPlaceHolder();
637     }
638 }
639
640
641 ////////////////////////////////////////////////////////////////////
642 //                        PropertyPlaceHolder                     //
643 ////////////////////////////////////////////////////////////////////
```

```
644
645 mapping SpringModel::PropertyPlaceholder::toPropertyPlaceHolder() :
    ExtendedSpringModel::springConfigDsl::PropertyPlaceholder {
646     propertyfile := self.propertyfile.map toPropertyFile();
647     ignoreResourceNotFound := self.ignoreResourceNotFound.getBoolean();
648     ignoreUnresolvable := self.ignoreUnresolvable.getBoolean();
649     localOverride := self.localOverride.getBoolean();
650     order := self.order;
651     propertiesRef := self.propertiesRef.map toComponent();
652     systemPropertiesMode := self.systemPropertiesMode;
653 }
654
655
656 ////////////////////////////////////////////////////////////////
657 //                      PropertyFile                          //
658 ////////////////////////////////////////////////////////////////
659
660 mapping SpringModel::PropertyFile::toPropertyFile() :
    ExtendedSpringModel::springConfigDsl::PropertyFile {
661     location := self.location;
662     fileEncoding := self.fileEncoding;
663 }
664
665
666 ////////////////////////////////////////////////////////////////
667 //                      SpringConfigured                      //
668 ////////////////////////////////////////////////////////////////
669
670 mapping SpringModel::SpringConfigured::toSpringConfigured() :
    ExtendedSpringModel::springConfigDsl::SpringConfigured {
671
672 }
673
674
675 ////////////////////////////////////////////////////////////////
676 //                      Aspect                                //
677 ////////////////////////////////////////////////////////////////
678
679 mapping SpringModel::Aspect::toAspect() : ExtendedSpringModel::springConfigDsl::Aspect {
680     if(self.oclIsTypeOf(SpringModel::AopAspectJAutoproxy))
681     {
682         self.oclAsType(SpringModel::AopAspectJAutoproxy).map toAopAspectJAutoproxy();
683     }
684     else
685     {
686         self.oclAsType(SpringModel::AopConfig).map toAopConfig();
687     }
688 }
689
690
691 ////////////////////////////////////////////////////////////////
692 //                      AopAspectJAutoproxy                   //
693 ////////////////////////////////////////////////////////////////
694
695 mapping SpringModel::AopAspectJAutoproxy::toAopAspectJAutoproxy() :
    ExtendedSpringModel::springConfigDsl::AopAspectJAutoproxy {
696     exposeProxy := self.exposeProxy.getBoolean();
697     proxyTrajetClass := self.proxyTrajetClass.getBoolean();
698     aopincludes := self.aopincludes->map toAopInclude();//getAopIncludes();
699 }
700
701
```

```
702 ////////////////////////////////////////////////////////////////
703 //                      AopInclude                            //
704 ////////////////////////////////////////////////////////////////
705
706 mapping SpringModel::AopInclude::toAopInclude() :
    ExtendedSpringModel::springConfigDsl::AopInclude {
707     aopInclude := self.aopInclude.map toComponent();
708 }
709
710
711 ////////////////////////////////////////////////////////////////
712 //                      AopConfig                             //
713 ////////////////////////////////////////////////////////////////
714
715 mapping SpringModel::AopConfig::toAopConfig() :
    ExtendedSpringModel::springConfigDsl::AopConfig {
716     exposeProxy := self.exposeProxy.getBoolean();
717     proxyTrajetClass := self.proxyTrajetClass.getBoolean();
718     aopPointcuts := self.aopPointcuts->map toAopPointCut();//getAopPointCuts();
719     aopAdvisors := self.aopAdvisors->map toAopAdvisor();//getAopAdvisors();
720     aspects := self.aspects->map toAopAspect();//getAopAspects();
721 }
722
723
724 ////////////////////////////////////////////////////////////////
725 //                      AopPointcut                           //
726 ////////////////////////////////////////////////////////////////
727
728 mapping SpringModel::AopPointcut::toAopPointCut() :
    ExtendedSpringModel::springConfigDsl::AopPointcut {
729     expression := self.expression;
730     name := self.name;
731 }
732
733
734 ////////////////////////////////////////////////////////////////
735 //                      AopAdvisor                            //
736 ////////////////////////////////////////////////////////////////
737
738 mapping SpringModel::AopAdvisor::toAopAdvisor() :
    ExtendedSpringModel::springConfigDsl::AopAdvisor {
739     adviceRef := self.adviceRef.map toTxAdvise();
740     name := self.name;
741     order := self.order;
742     pointcut := self.pointcut.map toAopPointCut();
743     pointcutRef := self.pointcutRef.map toAopPointCut();
744 }
745
746
747 ////////////////////////////////////////////////////////////////
748 //                      AopAspect                             //
749 ////////////////////////////////////////////////////////////////
750
751 mapping SpringModel::AopAspect::toAopAspect() :
    ExtendedSpringModel::springConfigDsl::AopAspect {
752     name := self.name;
753     order := self.order;
754     backingBeanRef := self.backingBeanRef.map toComponent();
755     aopPointcuts := self.aopPointcuts->map toAopPointCut();//getAopPointCuts();
756     declaredParents := self.declaredParents->map toDeclareParent();//getDeclaredParents();
757     advises := self.advises->map toAdvise();//getAdvises();
758 }
```

```
759
760
761 ////////////////////////////////////////////////////////////////
762 //                      TxAdvise                                //
763 ////////////////////////////////////////////////////////////////
764
765 mapping SpringModel::TxAdvise::toTxAdvise() :
    ExtendedSpringModel::springConfigDsl::TxAdvise {
766     name := self.name;
767     transactionManager := self.transactionManager;
768     txAttribute := self.txAttribute.map toTxAttribute();
769 }
770
771
772 ////////////////////////////////////////////////////////////////
773 //                      TxJtaTransactionManager                 //
774 ////////////////////////////////////////////////////////////////
775
776 mapping SpringModel::TxJtaTransactionManager::toTxJtaTransactionManager() :
    ExtendedSpringModel::springConfigDsl::TxJtaTransactionManager {
777
778 }
779
780
781 ////////////////////////////////////////////////////////////////
782 //                      TxAttribute                             //
783 ////////////////////////////////////////////////////////////////
784
785 mapping SpringModel::TxAttribute::toTxAttribute() :
    ExtendedSpringModel::springConfigDsl::TxAttribute {
786     txMethods := self.txMethods->map toTxMethod();//getTxMethods();
787 }
788
789
790 ////////////////////////////////////////////////////////////////
791 //                      TxMethod                                //
792 ////////////////////////////////////////////////////////////////
793
794 mapping SpringModel::TxMethod::toTxMethod() :
    ExtendedSpringModel::springConfigDsl::TxMethod {
795     name := self.name;
796     isolation := self.isolation.getIsolation();
797     noRollBackFor := self.noRollBackFor;
798     propagation := self.propagation.getIsolation();
799     readOnly := self.readOnly.getBoolean();
800     rollbackFor := self.rollbackFor;
801     timeOut := self.timeOut;
802 }
803
804
805 ////////////////////////////////////////////////////////////////
806 //                      TxAttribute                             //
807 ////////////////////////////////////////////////////////////////
808
809 mapping SpringModel::DeclareParents::toDeclareParent() :
    ExtendedSpringModel::springConfigDsl::DeclareParents {
810     typeMatching := self.typeMatching;
811     implementInterface := self.implementInterface.map toSInterface();
812     defaultImplInterface := self.defaultImplInterface.map toAopDefaultImplInterface();
813     delegateImplRef := self.delegateImplRef.map toAopDelegateImplRef();
814 }
815
```

```
816
817 ////////////////////////////////////////////////////////////////
818 //                      SpringModel::Interface                //
819 ////////////////////////////////////////////////////////////////
820
821 mapping SpringModel::Interface::toSInterface() :
    ExtendedSpringModel::springConfigDsl::Interface {
822     name := self.name;
823 }
824
825
826 ////////////////////////////////////////////////////////////////
827 //                      AopDefaultImplInterface               //
828 ////////////////////////////////////////////////////////////////
829
830 mapping SpringModel::AopDefaultImplInterface::toAopDefaultImplInterface() :
    ExtendedSpringModel::springConfigDsl::AopDefaultImplInterface {
831     name := self.name;
832 }
833
834
835 ////////////////////////////////////////////////////////////////
836 //                      AopDelegateImplRef                    //
837 ////////////////////////////////////////////////////////////////
838
839 mapping SpringModel::AopDelegateImplRef::toAopDelegateImplRef() :
    ExtendedSpringModel::springConfigDsl::AopDelegateImplRef {
840     ref := self.ref.map toComponent();
841 }
842
843
844 ////////////////////////////////////////////////////////////////
845 //                      Advise                                //
846 ////////////////////////////////////////////////////////////////
847
848 mapping SpringModel::Advise::toAdvise() : ExtendedSpringModel::springConfigDsl::Advise {
849     if(self.oclIsTypeOf(SpringModel::AfterReturning))
850     {
851         self.oclAsType(SpringModel::AfterReturning).map toAfterReturning();
852     }
853     else if(self.oclIsTypeOf(SpringModel::AfterThowing))
854     {
855         self.oclAsType(SpringModel::AfterThowing).map toAfterThrowing();
856     }
857     else
858     {
859         pointcutRef := self.pointcutRef.map toAopPointCut();
860         PointcutExpression := self.PointcutExpression;
861         method := self.method;
862     }
863 }
864
865
866 ////////////////////////////////////////////////////////////////
867 //                      AfterReturning                        //
868 ////////////////////////////////////////////////////////////////
869
870 mapping SpringModel::AfterReturning::toAfterReturning() :
    ExtendedSpringModel::springConfigDsl::AfterReturning {
871     pointcutRef := self.pointcutRef.map toAopPointCut();
872     PointcutExpression := self.PointcutExpression;
873     method := self.method;
```

```
874     returningValue := self.returningValue;
875 }
876
877
878 ////////////////////////////////////////////////////////////////
879 //                       AfterThowing                         //
880 ////////////////////////////////////////////////////////////////
881
882 mapping SpringModel::AfterThowing::toAfterThrowing() :
    ExtendedSpringModel::springConfigDsl::AfterThowing {
883     pointcutRef := self.pointcutRef.map toAopPointCut();
884     PointcutExpression := self.PointcutExpression;
885     method := self.method;
886     throwingValue := self.throwingValue;
887 }
888
889
890
891
892
893 ////////////////////////////////////////////////////////////////////////////
894 ////////////////////////////////////////////////////////////////////////////
895 //                                                                        //
896 //                               QUERIES                                  //
897 //                                                                        //
898 ////////////////////////////////////////////////////////////////////////////
899 ////////////////////////////////////////////////////////////////////////////
900
901
902 /* The following querries stand for returning the value of enumerated types. */
903
904
905 ////////////////////////////////////////////////////////////////
906 //                       getDefaultAutoWire                   //
907 ////////////////////////////////////////////////////////////////
908
909 query SpringModel::AutoWiredType::getDefaultAutoWire() :
    ExtendedSpringModel::springConfigDsl::AutoWiredType {
910     if(self.toString().equalsIgnoreCase(ExtendedSpringModel::springConfigDsl::AutoWiredTyp
    e::NO.toString()))
911         return ExtendedSpringModel::springConfigDsl::AutoWiredType::NO;
912     if(self.toString().equalsIgnoreCase(ExtendedSpringModel::springConfigDsl::AutoWiredTyp
    e::BYNAME.toString()))
913         return ExtendedSpringModel::springConfigDsl::AutoWiredType::BYNAME;
914     if(self.toString().equalsIgnoreCase(ExtendedSpringModel::springConfigDsl::AutoWiredTyp
    e::BYTYPE.toString()))
915         return ExtendedSpringModel::springConfigDsl::AutoWiredType::BYTYPE;
916     if(self.toString().equalsIgnoreCase(ExtendedSpringModel::springConfigDsl::AutoWiredTyp
    e::CONSTRUCTOR.toString()))
917         return ExtendedSpringModel::springConfigDsl::AutoWiredType::CONSTRUCTOR;
918     return ExtendedSpringModel::springConfigDsl::AutoWiredType::DEFAULT;
919 }
920
921
922 ////////////////////////////////////////////////////////////////
923 //                       getDefaultBoolean                    //
924 ////////////////////////////////////////////////////////////////
925
926 query SpringModel::DefaultableBoolean::getDefaultableBoolean() :
    ExtendedSpringModel::springConfigDsl::DefaultableBoolean {
927     if(self.toString().equalsIgnoreCase(ExtendedSpringModel::springConfigDsl::DefaultableB
    oolean::FALSE.toString()))
```

```
928         return ExtendedSpringModel::springConfigDsl::DefaultableBoolean::FALSE;
929     if(self.toString().equalsIgnoreCase(ExtendedSpringModel::springConfigDsl::DefaultableB
    oolean::TRUE.toString()))
930         return ExtendedSpringModel::springConfigDsl::DefaultableBoolean::TRUE;
931     return ExtendedSpringModel::springConfigDsl::DefaultableBoolean::DEFAULT;
932 }
933
934
935 ////////////////////////////////////////////////////////////////
936 //                        getBoolean                          //
937 ////////////////////////////////////////////////////////////////
938
939 query SpringModel::sBoolean::getBoolean() : ExtendedSpringModel::springConfigDsl::sBoolean
    {
940     if(self.toString().equalsIgnoreCase(ExtendedSpringModel::springConfigDsl::sBoolean::FA
    LSE.toString()))
941         return ExtendedSpringModel::springConfigDsl::sBoolean::FALSE;
942     return ExtendedSpringModel::springConfigDsl::sBoolean::TRUE;
943 }
944
945
946 ////////////////////////////////////////////////////////////////
947 //                        getScopedProxy                      //
948 ////////////////////////////////////////////////////////////////
949
950 query SpringModel::EnumScopedProxy::getScopedProxy() :
    ExtendedSpringModel::springConfigDsl::EnumScopedProxy {
951     if(self.toString().equalsIgnoreCase(ExtendedSpringModel::springConfigDsl::EnumScopedPr
    oxy::INTERFACES.toString()))
952     {
953         return ExtendedSpringModel::springConfigDsl::EnumScopedProxy::INTERFACES;
954     }
955     else
    if(self.toString().equalsIgnoreCase(ExtendedSpringModel::springConfigDsl::EnumScopedProxy:
    :TARGETCLASS.toString()))
956     {
957         return ExtendedSpringModel::springConfigDsl::EnumScopedProxy::TARGETCLASS;
958     }
959     else
960     {
961         return ExtendedSpringModel::springConfigDsl::EnumScopedProxy::NO;
962     }
963 }
964
965
966 ////////////////////////////////////////////////////////////////
967 //                        getTypeFilter                       //
968 ////////////////////////////////////////////////////////////////
969
970 query SpringModel::EnumTypeFilter::getTypeFilter() :
    ExtendedSpringModel::springConfigDsl::EnumTypeFilter {
971     if(self.toString().equalsIgnoreCase(ExtendedSpringModel::springConfigDsl::EnumTypeFilt
    er::ANNOTATION.toString()))
972     {
973         return ExtendedSpringModel::springConfigDsl::EnumTypeFilter::ANNOTATION;
974     }
975     else
    if(self.toString().equalsIgnoreCase(ExtendedSpringModel::springConfigDsl::EnumTypeFilter::
    ASSIGNABLE.toString()))
976     {
977         return ExtendedSpringModel::springConfigDsl::EnumTypeFilter::ASSIGNABLE;
978     }
```

```
979     else
    if(self.toString().equalsIgnoreCase(ExtendedSpringModel::springConfigDsl::EnumTypeFilter::
    ASPECTJ.toString()))
980     {
981         return ExtendedSpringModel::springConfigDsl::EnumTypeFilter::ASPECTJ;
982     }
983     else
    if(self.toString().equalsIgnoreCase(ExtendedSpringModel::springConfigDsl::EnumTypeFilter::
    REGEX.toString()))
984     {
985         return ExtendedSpringModel::springConfigDsl::EnumTypeFilter::REGEX;
986     }
987     else
988     {
989         return ExtendedSpringModel::springConfigDsl::EnumTypeFilter::CUSTOM;
990     }
991 }
992
993
994 ////////////////////////////////////////////////////////////////
995 //                      getMbeanRegistration                  //
996 ////////////////////////////////////////////////////////////////
997
998 query SpringModel::MbeanRegistrationEnum::getMbeanRegistration() :
    ExtendedSpringModel::springConfigDsl::MbeanRegistrationEnum {
999     if(self.toString().equalsIgnoreCase(ExtendedSpringModel::springConfigDsl::MbeanRegistr
    ationEnum::FAILONEXISTING.toString()))
1000    {
1001        return return
    ExtendedSpringModel::springConfigDsl::MbeanRegistrationEnum::FAILONEXISTING;
1002    }
1003    else
    if(self.toString().equalsIgnoreCase(ExtendedSpringModel::springConfigDsl::MbeanRegistratio
    nEnum::IGNOREEXISTING.toString()))
1004    {
1005        return
    ExtendedSpringModel::springConfigDsl::MbeanRegistrationEnum::IGNOREEXISTING;
1006    }
1007    else
1008    {
1009        return
    ExtendedSpringModel::springConfigDsl::MbeanRegistrationEnum::REPLACEEXISTING
1010    }
1011 }
1012
1013
1014 ////////////////////////////////////////////////////////////////
1015 //                      getIsolation                          //
1016 ////////////////////////////////////////////////////////////////
1017
1018 query SpringModel::EnumIsolation::getIsolation() :
    ExtendedSpringModel::springConfigDsl::EnumIsolation {
1019    if(self.toString().equalsIgnoreCase(ExtendedSpringModel::springConfigDsl::EnumIsolatio
    n::READ_UNCOMMITTED.toString()))
1020        return ExtendedSpringModel::springConfigDsl::EnumIsolation::READ_UNCOMMITTED;
1021    if(self.toString().equalsIgnoreCase(ExtendedSpringModel::springConfigDsl::EnumIsolatio
    n::READ_COMMITTED.toString()))
1022        return ExtendedSpringModel::springConfigDsl::EnumIsolation::READ_COMMITTED;
1023    if(self.toString().equalsIgnoreCase(ExtendedSpringModel::springConfigDsl::EnumIsolatio
    n::REPEATABLE_READ.toString()))
1024        return ExtendedSpringModel::springConfigDsl::EnumIsolation::REPEATABLE_READ;
1025    if(self.toString().equalsIgnoreCase(ExtendedSpringModel::springConfigDsl::EnumIsolatio
```

```
    n::SERIALIZABLE.toString()))
1026        return ExtendedSpringModel::springConfigDsl::EnumIsolation::SERIALIZABLE;
1027    return ExtendedSpringModel::springConfigDsl::EnumIsolation::DEFAULT;
1028 }
1029
1030
1031
1032
1033 ////////////////////////////////////////////////////////////////////////////////
1034 ////////////////////////////////////////////////////////////////////////////////
1035 //                                                                            //
1036 //                              COMMENTED QUERIES                             //
1037 //                                                                            //
1038 ////////////////////////////////////////////////////////////////////////////////
1039 ////////////////////////////////////////////////////////////////////////////////
1040
1041
1042 /*
1043 *
1044 *   The following querries return ordered sets
1045 *
1046 *   Those querries are left into the comments in case
1047 *   we need to ensure the order in sets
1048 *
1049 */
1050
1051
1052 ////////////////////////////////////////////////////////////////
1053 //                      getComponents                         //
1054 ////////////////////////////////////////////////////////////////
1055
1056 //query OrderedSet(SpringModel::Component)::getComponents() :
    OrderedSet(ExtendedSpringModel::Comp) {
1057 //  var comps : OrderedSet(ExtendedSpringModel::Comp);
1058 //  self->forEach(c)
1059 //  {
1060 //      comps += c.map toComponent()
1061 //  };
1062 //  return comps;
1063 //}
1064
1065
1066 ////////////////////////////////////////////////////////////////
1067 //                      getFeatures                           //
1068 ////////////////////////////////////////////////////////////////
1069
1070 //query OrderedSet(SpringModel::Feature)::getFeatures() :
    OrderedSet(ExtendedSpringModel::springConfigDsl::Feature) {
1071 //  var features : OrderedSet(ExtendedSpringModel::springConfigDsl::Feature);
1072 //  self->forEach(f)
1073 //  {
1074 //      features += f.map toFeature();
1075 //  };
1076 //  return features;
1077 //}
1078
1079
1080 ////////////////////////////////////////////////////////////////
1081 //                      getAbstractArtefacts                  //
1082 ////////////////////////////////////////////////////////////////
1083
1084
```

```
1085 //query OrderedSet(SpringModel::AbstractArtefact)::getAbstractArtefacts() :
     OrderedSet(ExtendedSpringModel::springConfigDsl::AbstractArtefact) {
1086 //  var artefacts : OrderedSet(ExtendedSpringModel::springConfigDsl::AbstractArtefact);
1087 //  self->forEach(a)
1088 //  {
1089 //      artefacts += a.map toAbstractArtefact();
1090 //  };
1091 //  return artefacts;
1092 //}
1093
1094
1095 ////////////////////////////////////////////////////////////////
1096 //                      getProps                              //
1097 ////////////////////////////////////////////////////////////////
1098
1099 //query OrderedSet(SpringModel::Prop)::getProps() :
     OrderedSet(ExtendedSpringModel::springConfigDsl::Prop) {
1100 //  var props : OrderedSet(ExtendedSpringModel::springConfigDsl::Prop);
1101 //  self->forEach(p)
1102 //  {
1103 //      props += p.map toProp();
1104 //  };
1105 //  return props;
1106 //}
1107
1108
1109 ////////////////////////////////////////////////////////////////
1110 //                      getMapEntries                         //
1111 ////////////////////////////////////////////////////////////////
1112
1113 //query OrderedSet(SpringModel::MapEntry)::getMapEntries() :
     OrderedSet(ExtendedSpringModel::springConfigDsl::MapEntry) {
1114 //  var mapEntries : OrderedSet(ExtendedSpringModel::springConfigDsl::MapEntry);
1115 //  self->forEach(me)
1116 //  {
1117 //      mapEntries += me.map toMapEntry();
1118 //  };
1119 //  return mapEntries;
1120 //}
1121
1122
1123 ////////////////////////////////////////////////////////////////
1124 //                      getLookupMethods                      //
1125 ////////////////////////////////////////////////////////////////
1126
1127 //query OrderedSet(SpringModel::LookupMethod)::getLookupMethods() :
     OrderedSet(ExtendedSpringModel::springConfigDsl::LookupMethod) {
1128 //  var lookupMethods : OrderedSet(ExtendedSpringModel::springConfigDsl::LookupMethod);
1129 //  self->forEach(lm)
1130 //  {
1131 //      lookupMethods += lm.map toLookupMethod();
1132 //  };
1133 //  return lookupMethods;
1134 //}
1135
1136
1137 ////////////////////////////////////////////////////////////////
1138 //                      getQualifiers                         //
1139 ////////////////////////////////////////////////////////////////
1140
1141 //query OrderedSet(SpringModel::Qualifier)::getQualifiers() :
     OrderedSet(ExtendedSpringModel::springConfigDsl::Qualifier) {
```

```
1142 //  var qualifiers : OrderedSet(ExtendedSpringModel::springConfigDsl::Qualifier);
1143 //  self->forEach(q)
1144 //  {
1145 //      qualifiers += q.map toQualifier();
1146 //  };
1147 //  return qualifiers;
1148 //}
1149
1150
1151 ////////////////////////////////////////////////////////////////
1152 //                      getQualifierAttributes                //
1153 ////////////////////////////////////////////////////////////////
1154
1155 //query OrderedSet(SpringModel::QualifierAttribute)::getQualifierAttributes() :
        OrderedSet(ExtendedSpringModel::springConfigDsl::QualifierAttribute) {
1156 //  var qualifierAttributes :
        OrderedSet(ExtendedSpringModel::springConfigDsl::QualifierAttribute);
1157 //  self->forEach(qa)
1158 //  {
1159 //      qualifierAttributes += qa.map toQualifierAttribute();
1160 //  };
1161 //  return qualifierAttributes;
1162 //}
1163
1164
1165 ////////////////////////////////////////////////////////////////
1166 //                      getMetas                              //
1167 ////////////////////////////////////////////////////////////////
1168
1169 //query OrderedSet(SpringModel::Meta)::getMetas() :
        OrderedSet(ExtendedSpringModel::springConfigDsl::Meta) {
1170 //  var metas : OrderedSet(ExtendedSpringModel::springConfigDsl::Meta);
1171 //  self->forEach(m)
1172 //  {
1173 //      metas += m.map toMeta();
1174 //  };
1175 //  return metas;
1176 //}
1177
1178
1179 ////////////////////////////////////////////////////////////////
1180 //                      getUtilPropertiesPath                 //
1181 ////////////////////////////////////////////////////////////////
1182
1183 //query OrderedSet(SpringModel::UtilPropertyPath)::getUtilPropertiesPath() :
        OrderedSet(ExtendedSpringModel::springConfigDsl::UtilPropertyPath) {
1184 //  var utilPropertiesPath :
        OrderedSet(ExtendedSpringModel::springConfigDsl::UtilPropertyPath);
1185 //  self->forEach(upp)
1186 //  {
1187 //      utilPropertiesPath += upp.map toUtilPropertyPath();
1188 //  };
1189 //  return utilPropertiesPath;
1190 //}
1191
1192
1193 ////////////////////////////////////////////////////////////////
1194 //                      getAliases                            //
1195 ////////////////////////////////////////////////////////////////
1196
1197 //query OrderedSet(SpringModel::Alias)::getAliases() :
        OrderedSet(ExtendedSpringModel::springConfigDsl::Alias) {
```

```
1198 //  var aliases : OrderedSet(ExtendedSpringModel::springConfigDsl::Alias);
1199 //  self->forEach(a)
1200 //  {
1201 //      aliases += a.map toAlias();
1202 //  };
1203 //  return aliases;
1204 //}
1205
1206
1207 ////////////////////////////////////////////////////////////////
1208 //                      getImports                            //
1209 ////////////////////////////////////////////////////////////////
1210
1211 //query OrderedSet(SpringModel::Import)::getImports() :
     OrderedSet(ExtendedSpringModel::springConfigDsl::Import) {
1212 //  var imports : OrderedSet(ExtendedSpringModel::springConfigDsl::Import);
1213 //  self->forEach(i)
1214 //  {
1215 //      imports += i.map toImport();
1216 //  };
1217 //  return imports;
1218 //}
1219
1220
1221 ////////////////////////////////////////////////////////////////
1222 //                      getContexts                           //
1223 ////////////////////////////////////////////////////////////////
1224
1225 //query OrderedSet(SpringModel::Context)::getContexts() :
     OrderedSet(ExtendedSpringModel::springConfigDsl::Context) {
1226 //  var contexts : OrderedSet(ExtendedSpringModel::springConfigDsl::Context);
1227 //  self->forEach(c)
1228 //  {
1229 //      contexts += c.map toContext();
1230 //  };
1231 //  return contexts;
1232 //}
1233
1234
1235 ////////////////////////////////////////////////////////////////
1236 //                      getIncludeFilters                     //
1237 ////////////////////////////////////////////////////////////////
1238
1239 //query OrderedSet(SpringModel::IncludeFilter)::getIncludeFilters() :
     OrderedSet(ExtendedSpringModel::springConfigDsl::IncludeFilter) {
1240 //  var includeFilters : OrderedSet(ExtendedSpringModel::springConfigDsl::IncludeFilter);
1241 //  self->forEach(iF)
1242 //  {
1243 //      includeFilters += iF.map toIncludeFilter();
1244 //  };
1245 //  return includeFilters;
1246 //}
1247
1248
1249 ////////////////////////////////////////////////////////////////
1250 //                      getExcludeFilters                     //
1251 ////////////////////////////////////////////////////////////////
1252
1253 //query OrderedSet(SpringModel::ExcludeFilter)::getExcludeFilters() :
     OrderedSet(ExtendedSpringModel::springConfigDsl::ExcludeFilter) {
1254 //  var excludeFilters : OrderedSet(ExtendedSpringModel::springConfigDsl::ExcludeFilter);
1255 //  self->forEach(eF)
```

```
1256 //  {
1257 //      excludeFilters += eF.map toExcludeFilter();
1258 //  };
1259 //  return excludeFilters;
1260 //}
1261
1262
1263 ////////////////////////////////////////////////////////////////
1264 //                      getAspects                            //
1265 ////////////////////////////////////////////////////////////////
1266
1267 //query OrderedSet(SpringModel::Aspect)::getAspects() :
     OrderedSet(ExtendedSpringModel::springConfigDsl::Aspect) {
1268 //  var aspects : OrderedSet(ExtendedSpringModel::springConfigDsl::Aspect);
1269 //  self->forEach(a)
1270 //  {
1271 //      aspects += a.map toAspect();
1272 //  };
1273 //  return aspects;
1274 //}
1275
1276
1277 ////////////////////////////////////////////////////////////////
1278 //                      getAopIncludes                        //
1279 ////////////////////////////////////////////////////////////////
1280
1281 //query OrderedSet(SpringModel::AopInclude)::getAopIncludes() :
     OrderedSet(ExtendedSpringModel::springConfigDsl::AopInclude) {
1282 //  var aopIncludes : OrderedSet(ExtendedSpringModel::springConfigDsl::AopInclude);
1283 //  self->forEach(ai)
1284 //  {
1285 //      aopIncludes += ai.map toAopInclude();
1286 //  };
1287 //  return aopIncludes;
1288 //}
1289
1290
1291 ////////////////////////////////////////////////////////////////
1292 //                      getAopPointCuts                       //
1293 ////////////////////////////////////////////////////////////////
1294
1295 //query OrderedSet(SpringModel::AopPointcut)::getAopPointCuts() :
     OrderedSet(ExtendedSpringModel::springConfigDsl::AopPointcut) {
1296 //  var aopPointcuts : OrderedSet(ExtendedSpringModel::springConfigDsl::AopPointcut);
1297 //  self->forEach(ap)
1298 //  {
1299 //      aopPointcuts += ap.map toAopPointCut();
1300 //  };
1301 //  return aopPointcuts;
1302 //}
1303
1304
1305 ////////////////////////////////////////////////////////////////
1306 //                      getAopAdvisors                        //
1307 ////////////////////////////////////////////////////////////////
1308
1309 //query OrderedSet(SpringModel::AopAdvisor)::getAopAdvisors() :
     OrderedSet(ExtendedSpringModel::springConfigDsl::AopAdvisor) {
1310 //  var aopAdvisors : OrderedSet(ExtendedSpringModel::springConfigDsl::AopAdvisor);
1311 //  self->forEach(aa)
1312 //  {
1313 //      aopAdvisors += aa.map toAopAdvisor();
```

```
1314 //  };
1315 //  return aopAdvisors;
1316 //}
1317
1318
1319 ////////////////////////////////////////////////////////////////
1320 //                      getTxMethods                          //
1321 ////////////////////////////////////////////////////////////////
1322
1323 //query OrderedSet(SpringModel::TxMethod)::getTxMethods() :
     OrderedSet(ExtendedSpringModel::springConfigDsl::TxMethod) {
1324 //  var txMethods : OrderedSet(ExtendedSpringModel::springConfigDsl::TxMethod);
1325 //  self->forEach(tm)
1326 //  {
1327 //      txMethods += tm.map toTxMethod();
1328 //  };
1329 //  return txMethods;
1330 //}
1331
1332
1333 ////////////////////////////////////////////////////////////////
1334 //                      getAopAspects                         //
1335 ////////////////////////////////////////////////////////////////
1336
1337 //query OrderedSet(SpringModel::AopAspect)::getAopAspects() :
     OrderedSet(ExtendedSpringModel::springConfigDsl::AopAspect) {
1338 //  var aopAspects : OrderedSet(ExtendedSpringModel::springConfigDsl::AopAspect);
1339 //  self->forEach(aa)
1340 //  {
1341 //      aopAspects += aa.map toAopAspect();
1342 //  };
1343 //  return aopAspects;
1344 //}
1345
1346
1347 ////////////////////////////////////////////////////////////////
1348 //                      getDeclaredParents                    //
1349 ////////////////////////////////////////////////////////////////
1350
1351 //query OrderedSet(SpringModel::DeclareParents)::getDeclaredParents() :
     OrderedSet(ExtendedSpringModel::springConfigDsl::DeclareParents) {
1352 //  var declaredParents :
     OrderedSet(ExtendedSpringModel::springConfigDsl::DeclareParents);
1353 //  self->forEach(dp)
1354 //  {
1355 //      declaredParents += dp.map toDeclareParent();
1356 //  };
1357 //  return declaredParents;
1358 //}
1359
1360
1361 ////////////////////////////////////////////////////////////////
1362 //                      getAdvises                            //
1363 ////////////////////////////////////////////////////////////////
1364
1365 //query OrderedSet(SpringModel::Advise)::getAdvises() :
     OrderedSet(ExtendedSpringModel::springConfigDsl::Advise) {
1366 //  var advises : OrderedSet(ExtendedSpringModel::springConfigDsl::Advise);
1367 //  self->forEach(a)
1368 //  {
1369 //      advises += a.map toAdvise();
1370 //  };
```

```
1371 //  return advises;
1372 //}
1373
1374
1375 ////////////////////////////////////////////////////////////////
1376 //                      getTxAdvices                          //
1377 ////////////////////////////////////////////////////////////////
1378
1379 //query OrderedSet(SpringModel::TxAdvise)::getTxAdvices() :
     OrderedSet(ExtendedSpringModel::springConfigDsl::TxAdvise) {
1380 //  var txAdvices : OrderedSet(ExtendedSpringModel::springConfigDsl::TxAdvise);
1381 //  self->forEach(ta)
1382 //  {
1383 //      txAdvices += ta.map toTxAdvise();
1384 //  };
1385 //  return txAdvices;
1386 //}
```