Models

Why DOM?

- XML means more than is written there
- We want to work with semantic model
- XML PSI is not very convenient

plugin.xml:

```
<idea-plugin>
  <id>com.intellij.intelliWhisper</id>
  ...
  <extension-points>
    <extension-point name="voice"
        interface="com.intellij.whisper.Voice"/>
    ...
  </extension-points>
  ...
</idea-plugin>
```

Extension points' qualified names

```
IdeaPlugin root = ...;
Map<String, PsiClass> result = new HashMap<~>();
String id = root.getId().getValue();
if (id != null) {
  EPGroup group = root.getEPGroup();
  for (ExtensionPoint ep : group.getExtensionPoints()) {
    String name = ep.getName().getValue();
    if (name != null) {
      result.put(id + "." + name,
                 ep.getInterface().getValue());
    }
  }
}
```

DOM interfaces

```
interface IdeaPlugin extends DomElement {
  GenericDomValue<String> getId();

  @SubTag("extension-points")
  EPGroup getEPGroup();
}

interface EPGroup extends DomElement {
  List<ExtensionPoint> getExtensionPoints();
}

interface ExtensionPoint extends DomElement {
  GenericAttributeValue<String> getName();

  GenericAttributeValue<PsiClass> getInterface();
}
```

Advantages

- Auto-generated implementations
- Easy reference registration
- Easy declaration registration
- Runtime extensibility
- Optionally: completion for tag & attribute names
- http://www.jetbrains.com/idea/documentation/dom-rp.html

SEM

- Compute something by PsiElement and cache it
- Drop the cache on any change
- SemService#getSemElement(SemKey, PsiElement)
- registration: SemContributor
- register(SemKey<Sem>, ElementPattern<Psi>, NullableFunction<Psi,Sem>)
- No side effects, please
- equals/hashCode


ДЖЕМ

JAM

- Semantics based on Java annotations, method signatures, etc.
- PsiRef<Psi>: real || imaginary
- Custom wrappers for members and nested annotations
- Bound to JamService.JAM_ELEMENT_KEY
- Domain-specific logic inside wrappers
- JAM children

Meta

- Registering nested PsiMember JAM considered superfluous
- JamService.MEMBER_META_KEY
- Instantiates the JAM element
- Generates proxy class
- Knows about children: JamChildrenQuery<ChildJam>

AOP aspects & introductions

```
JamChildrenQuery<AopIntroductionImpl> INTRODUCTIONS_QUERY =
    annotatedFields (DECLARE_PARENTS_ANNO , AopIntroductionImpl.class);

JamClassMeta<AopAspectImpl> ASPECT_META =
    new JamClassMeta<~>(AopAspectImpl.class).
    addChildrenQuery(INTRODUCTIONS_QUERY );
....

ASPECT_META .register (registrar,
 psiClass ().
   withAnnotation(ASPECT_ANNO ).
   andNot(psiElement ().compiled()));
....

jamService.getJamClassElements(AopAspectImpl.class, ASPECT_ANNO , scope)
....

public List<AopIntroductionImpl> getIntroductions() {
   return INTRODUCTIONS_QUERY .findChildren(getClassRef());
}
```

Attributes

- Annotation meta
- Attribute meta: class, enum, annotation, string
- Single or array
- JamStringAttributeElement<T>: T getValue()
- JamConverter<T>: T fromString(...), createReferences(...)

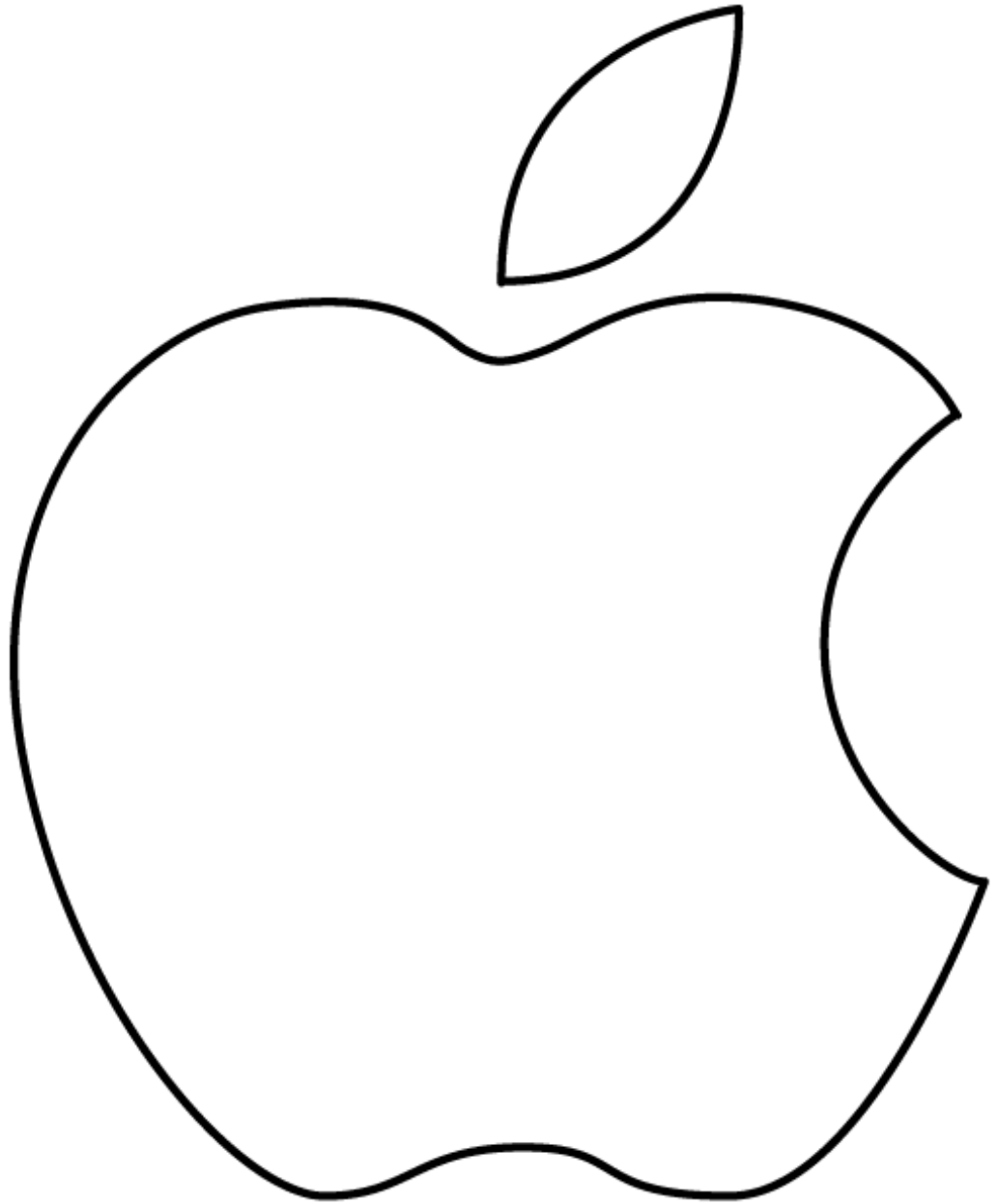
AOP: throwing attribute

```
JamStringAttributeMeta.Single<PsiParameter> THROWING_META =
   singleString(THROWING_PARAM , MethodParameterConverter.INSTANCE );

JamAnnotationMeta AFTER_THROWING_META =
  new JamAnnotationMeta(AFTER_THROWING_ANNO ).addAttribute(THROWING_META );

....

public JamStringAttributeElement<PsiParameter> getReturning() {
  return AFTER_THROWING_META .getAttribute(getMethod(), THROWING_META );
}
```



POM

- Need to resolve somewhere not in PSI
- FakePsiElement, RenameableFakePsiElement
- interface PomTarget extends Navigatable {}
- PomNamedTarget, PsiTarget
- Register PomTarget's with JamMemberMeta
- JamPomTargets on string attribute