PHP Open API

WORK IN PROGRESS.

Using php-openapi

```
<depends>com.jetbrains.php</depends>
```

PHP PSI

com.jetbrains.php.lang.psi.elements.*;

Utility Classes

PHP Extension Points

PhpTypeProvider

Here is a code fragment that makes generic Factory Method support work

```
<php.typeProvider3
implementation="com.jetbrains.php.lang.psi.resolve.types.PhpStaticFactoryTypeProvider3"/>
```

Interface:

```
/**
 * Extension point to implement to provide Type information on various PhpPsiElements.
 */
public interface PhpTypeProvider3 {

  ExtensionPointName<PhpTypeProvider3> EP_NAME =
ExtensionPointName.create("com.jetbrains.php.typeProvider3");

  /**
   * @return  Your custom signature key, i.e. "Я". Do not use any of PhpTypeSignatureKey.XXX
constants though!
   */
  char getKey();

  /**
   * @param element to deduce type for - using only LOCAL info. <b>THIS IS MOST CRUCIAL ASPECT
TO FOLLOW</b>
   * @return type for element, null if no insight. You can return a custom signature here to be later
decoded by method below.
   */
  @Nullable
  PhpType getType(PsiElement element);

  /**
   * Here you can extend the signature lookups
   * @param expression Signature expression to decode. You can use PhpIndex.getBySignature() to
look up expression internals.
   * @param visited Recursion guard: please pass this on into any phpIndex calls having same
parameter
   * @param depth Recursion guard: please pass this on into any phpIndex calls having same
parameter
   * @param project well so you can reach the PhpIndex
   * @return null if no match
   */
  Collection<? extends PhpNamedElement> getBySignature(String expression, Set<String> visited, int
depth, Project project);
}
```

Our implementation: includes a Completion contributor for the parameter values too.

```
/**
 */
public class PhpStaticFactoryTypeProvider extends CompletionContributor implements
PhpTypeProvider3 {

  private static final Key<CachedValue<Map<String, Map<String, String>>>>
STATIC_FACTORY_TYPE_MAP =
    new Key<CachedValue<Map<String, Map<String, String>>>>("STATIC_FACTORY_TYPE_MAP");

  @Override
  public char getKey() {
    return 'S';
  }

  @Nullable
  @Override
```

```
  public String getType(PsiElement e) {
    if (e instanceof MethodReference && ((MethodReference)e).isStatic()) {
      Map<String, Map<String, String>> methods = getStaticMethodTypesMap(e.getProject());
      String refSignature = ((MethodReference)e).getSignature();
      if (methods.containsKey(refSignature)) {
        PsiElement[] parameters = ((MethodReference)e).getParameters();
        if (parameters.length > 0) {
          PsiElement parameter = parameters[0];
          if (parameter instanceof StringLiteralExpression) {
            String param = ((StringLiteralExpression)parameter).getContents();
            if (StringUtil.isNotEmpty(param)) return refSignature + "." + param;
          }
        }
      }
    }
    return null;
  }

  @Override
  public Collection<? extends PhpNamedElement> getBySignature(String expression, Project project) {
    Map<String, Map<String, String>> methods = getStaticMethodTypesMap(project);
    int dot = expression.lastIndexOf('.');
    String refSignature = expression.substring(0, dot);
    Map<String, String> types = methods.get(refSignature);
    if (types != null) {
      String type = types.get(expression.substring(dot + 1));
      if (type != null) return PhpIndex.getInstance(project).getAnyByFQN(type);
    }
    return Collections.emptySet();
  }

  synchronized Map<String, Map<String, String>> getStaticMethodTypesMap(final Project project) {
    CachedValue<Map<String, Map<String, String>>> myStaticMethodTypesMap =
project.getUserData(STATIC_FACTORY_TYPE_MAP);
    if (myStaticMethodTypesMap == null) {
      myStaticMethodTypesMap = CachedValuesManager.getManager(project).createCachedValue(
        new CachedValueProvider<Map<String, Map<String, String>>>() {
          @Nullable
          @Override
          public Result<Map<String, Map<String, String>>> compute() {
            Map<String, Map<String, String>> map = new THashMap<String, Map<String, String>>();
            Collection<Variable> variables = getVariables(project, "STATIC_METHOD_TYPES");
            for (Variable variable : variables) {
              if (!"\\PHPSTORM_META\\".equals(variable.getNamespaceName())) continue;
              PsiElement parent = variable.getParent();
              if (parent instanceof AssignmentExpression) {
                PhpPsiElement value = ((AssignmentExpression)parent).getValue();
                if (value instanceof ArrayCreationExpression) {
                  for (ArrayHashElement element : ((ArrayCreationExpression)value).getHashElements()) {
                    PhpPsiElement match = element.getKey();
                    if (match instanceof MethodReference) {
                      String matchSignature = ((MethodReference)match).getSignature();
                      Map<String, String> types = map.get(matchSignature);
                      if (types == null) {
                        types = new THashMap<String, String>();
                        map.put(matchSignature, types);
                      }
                      PhpPsiElement val = element.getValue();
                      if (val instanceof ArrayCreationExpression) {
```

```
                    PhpPsiElement child = val.getFirstPsiChild();
                    while (child != null) {
                      if (child.getFirstPsiChild() instanceof BinaryExpression) {
                        BinaryExpression binary = ((BinaryExpression)child.getFirstPsiChild());
                        if (binary.getOperation().getNode().getElementType() ==
PhpTokenTypes.kwINSTANCEOF) {
                          PsiElement leftOperand = binary.getLeftOperand();
                          PsiElement rightOperand = binary.getRightOperand();
                          if (leftOperand instanceof StringLiteralExpression && rightOperand != null) {
                            types.put(((StringLiteralExpression)leftOperand).getContents(),
rightOperand.getText());
                          }
                        }
                      }
                      child = child.getNextPsiSibling();
                    }
                  }
                }
              }
            }
          }
          return CachedValueProvider.Result.create(map, getMetaFile(project));
        }
      }, false);
    project.putUserData(STATIC_FACTORY_TYPE_MAP, myStaticMethodTypesMap);
  }
  return myStaticMethodTypesMap.getValue();
 }

 private static Collection<Variable> getVariables(Project project, final String key) {
  PsiFile file = getMetaFile(project);
  final Collection<Variable> result = new SmartList<Variable>();
  if (file instanceof PhpFile) {
    //AG not the most elegant way - but still an allowed usage.
    //noinspection deprecation
    file.accept(new PhpRecursiveElementVisitor() {
      @Override
      public void visitPhpAssignmentExpression(AssignmentExpression assignmentExpression) {
        PhpPsiElement variable = assignmentExpression.getVariable();
        if (variable instanceof Variable) {
          if (((Variable)variable).getNameCS().equals(key)) {
            result.add((Variable)variable);
          }
        }
      }
    });
  }
  return result;
 }

 public static PsiFile getMetaFile(Project project) {
  VirtualFile metaFile = LocalFileSystem.getInstance().findFileByPath(project.getBasePath() +
File.separatorChar + ".phpstorm.meta.php");
  return metaFile != null ? PsiManager.getInstance(project).findFile(metaFile) : null;
 }

 @Override
 public void fillCompletionVariants(CompletionParameters parameters, CompletionResultSet result) {
```

```
    final ProcessingContext context = new ProcessingContext();
    PsiElement position = parameters.getPosition();
    if (parameters.getCompletionType() == CompletionType.BASIC &&
        psiElement().withParent(StringLiteralExpression.class).withSuperParent(2, ParameterList.class)
          .accepts(position, context)) {
      ParameterListOwner parameterListOwner = PsiTreeUtil.getStubOrPsiParentOfType(position,
ParameterListOwner.class);
      if (parameterListOwner instanceof MethodReference &&
((MethodReference)parameterListOwner).isStatic()) {

        Map<String, String> map =
getStaticMethodTypesMap(position.getProject()).get(((MethodReference)parameterListOwner).getSign
ature());
        for (String s : map.keySet()) {
          result.addElement(LookupElementBuilder.create(s).appendTailText(map.get(s), true));
        }
      }
    }
    super.fillCompletionVariants(parameters, result);
```

```
  }
}
```

to make completion work registration is required:

```
<completion.contributor language="PHP"

implementationClass="com.jetbrains.php.lang.psi.resolve.types.PhpStaticFactoryTypeProvider"/>
```