YAMF - Yet Another Model Factory

This functionality has been renamed Sling Models and is documented at http://sling.apache.org/documentation/bundles/models.html

Many Sling projects want to be able to create model objects - POJOs which are automatically mapped from Sling objects, typically resources, but also request objects. Sometimes these POJOs need OSGi services as well.

YAMF is an attempt to consolidate the various approaches I have seen to this problem.

Design Goals

- Entirely annotation driven. "Pure" POJOs.
- Use standard annotations where possible.
- Pluggable
- OOTB, support resource properties (via ValueMap), SlingBindings, OSGi services, request attributes
- Adapt multiple objects - minimal required Resource and SlingHttpServletRequest
- Client doesn't know/care that YAMF is involved
- Support both classes and interfaces.
- Work with existing Sling infrastructure (i.e. not require changes to other bundles).

Basic Usage

In the simplest case, the class is annotated with @Model and the adaptable class. Fields which need to be injected are annotated with @Inject:

```
@Model(adaptables=Resource.class)
public class MyModel {

    @Inject
    private String propertyName;
}
```

In this case, a property named 'propertyName' will be looked up from the Resource (after first adapting it to a ValueMap) and it is injected.

For an interface, it is similar:

```
@Model(adaptables=Resource.class)
public interface MyModel {

    @Inject
    String getPropertyName();
}
```

In order for these classes to be picked up, there is a header which must be added to the bundle's manifest:

```
<Sling-YAMF-Packages>
  org.apache.sling.yamf.it.models
</Sling-YAMF-Packages>
```

Client Code

Client code doesn't need to be aware that YAMF is being used. It just uses the Sling Adapter framework:

```
MyModel model = resource.adaptTo(MyModel.class)
```

Or

```
<sling:adaptTo adaptable="${resource}" adaptTo="org.apache.sling.yamf.it.models.MyModel" var="model"/>
```

Or

```
${sling:adaptTo(resource, 'org.apache.sling.yamf.it.models.MyModel')}
```

As with other AdapterFactories, if the adaptation can't be made for any reason, adaptTo() returns null.

Other Options

If the field or method name doesn't exactly match the property name, @Named can be used:

```
@Model(adaptables=Resource.class)
public class MyModel {

    @Inject @Named("secondPropertyName")
    private String otherName;
}
```

@Injected fields/methods are assumed to be required. To mark them as optional, use @Optional:

```
@Model(adaptables=Resource.class)
public class MyModel {

    @Inject @Optional
    private String otherName;
}
```

A default value can be provided (for Strings & primitives):

```
@Model(adaptables=Resource.class)
public class MyModel {

    @Inject @Default(values="defaultValue")
    private String name;
}
```

Defaults can also be arrays:

```
@Model(adaptables=Resource.class)
public class MyModel {

    @Inject @Default(intValues={1,2,3,4})
    private int[] integers;
}
```

OSGi services can be injected:

```
@Model(adaptables=Resource.class)
public class MyModel {

    @Inject
    private ResourceResolverFactory resourceResolverFactory;
}
```

In this case, the name is not used -- only the class name. Lists and arrays are supported:

```
@Model(adaptables=Resource.class)
public class MyModel {

    @Inject
    private List<Servlet> servlets;
}
```

OSGi injection can be filtered:

```
@Model(adaptables=SlingHttpServletRequest.class)
public class MyModel {

    @Inject
    private PrintWriter out;

    @Inject
    @Named("log")
    private Logger logger;

    @Inject
    @Filter("(paths=/bin/something)")
    private List<Servlet> servlets;
}
```

The @PostConstruct annotation can be used to add methods which are invoked upon completion of all injections:

```
@Model(adaptables=SlingHttpServletRequest.class)
public class MyModel {

    @Inject
    private PrintWriter out;

    @Inject
    @Named("log")
    private Logger logger;

    @PostConstruct
    protected void sayHello() {
         logger.info("hello");
    }
}
```

@PostConstruct methods in a super class will be invoked first.

If the injection should be based on a JavaBean property of the adaptable, you can indicate this using the @Via annotation:

```
@Model(adaptables=SlingHttpServletRequest.class)
public interface MyModel {

    // will return request.getResource().adaptTo(ValueMap.class).get("propertyName", String.class)
    @Inject @Via("resource")
    String getPropertyName();
}
```

If there is ambiguity where a given injection could be handled by more than one injector, the @Source annotation can be used to define which injector is responsible:

```
@Model(adaptables=SlingHttpServletRequest.class)
public interface MyModel {

    // Ensure that "resource" is retrived from the bindings, not a request attribute
    @Inject @Source("script-bindings")
    Resource getResource();
}
```

If the injected object does not match the desired type and the object implements the Adaptable interface, YAMF will try to adapt it. This provides the ability to create rich object graphs. For example:

```
@Model(adaptables=Resource.class)
public interface MyModel {

    @Inject
    ImageModel getImage();
}

@Model(adaptables=Resource.class)
public interface ImageModel {

    @Inject
    String getPath();
}
```

When a resource is adapted to MyModel, a child resource named image is automatically adapted to an instance of ImageModel.

Annotation Reference

- @Model - declares a model class or interface
- @Inject - marks a field or method as injectable
- @Named - declare a name for the injection (otherwise, defaults based on field or method name).
- @Optional - marks a field or method injection as optional
- @Source - explictly tie an injected field or method to a particular injector (by name). Can also be on other annotations.
- @Filter - an OSGi service filter
- @PostConstruct - methods to call upon model option creation (only for model classes)
- @Via - use a JavaBean property of the adaptable as the source of the injection
- @Default - set default values for a field or method

Available Injectors

Title	Name (for @Source)	Description	Applicable To (including using @Via)	Notes
Value Map	valuemap	Gets a property from a Value Map	Any object which is or can be adapted to a ValueMap	
OSGI Services	osgi-services	Lookup services based on class name	Any object	Effectively ignores name.
Script Bindings	script-bindings	Lookup objects in the script bindings object	A ServletRequest object which has the Sling Bindings attribute defined	
Child Resources	child-resources	Gets a child resource by name	Resource objects	
Request Attributes	request-attributes	Get a request attribute	ServletRequest objects	