APACHE SLING & FRIENDS TECH MEETUP
BERLIN, 25-27 SEPTEMBER 2017

Fun Times with Sling Models Exporters

Justin Edelson, Adobe Systems

Sling Models – Reducing Boilerplate Since ‘14

- OG
  - AdapterFactories

adaptTo()

Roadmap from 2014

- More Standard Injectors ✔
  - SelfInjector, SlingObjectInjector
- AEM-specific injectors in ACS AEM Commons ✔
  - AemObjectsInjector
- Pluggable @Via Support ✔
  - Done this year

Der Mensch denkt und Gott lenkt

- Injector-Specific Annotations
- ModelFactory
- Sling Validation Integration
- Alternate Adapter Classes
- BND Plugin
- Exporter Framework

Sling Models – Reducing Boilerplate Since ‘14

- OG
  - AdapterFactories
- Exporter Framework
  - Format Conversion
  - Content Access Servlets

Use Cases

- Customized content output for external applications
- Isomorphic rendering – use the same model on the client and server

In a Nutshell

- GET /content/something.model.json

1. Ada[illegible]ce to [illegible]bject
2. Transform t[illegible]ring
3. Serve t[illegible]

No Code Required*

How To

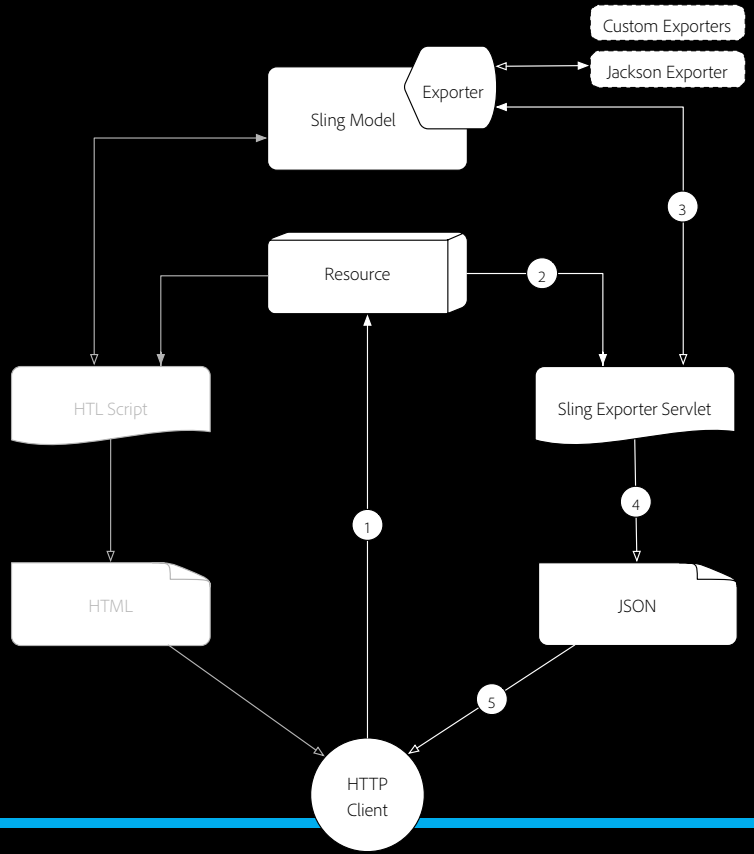
1. Add resourceType attribute to @Model
2. Add @Exporter(name = “jackson”, extension = “json”)
3. (Optional) Annotate model methods/properties.

How It Works

1. HTTP GET Request is made for a resource in AEM with the selector and extension registered with the Sling Model's Exporter.

   Example: *HTTP GET /content/my-resource.model.json*

2. Sling resolves the the requested resource's *sling:resourceType*, *selector* and *extension* to a dynamically generated Sling Exporter Servlet, which is mapped to the Sling Model with Exporter.



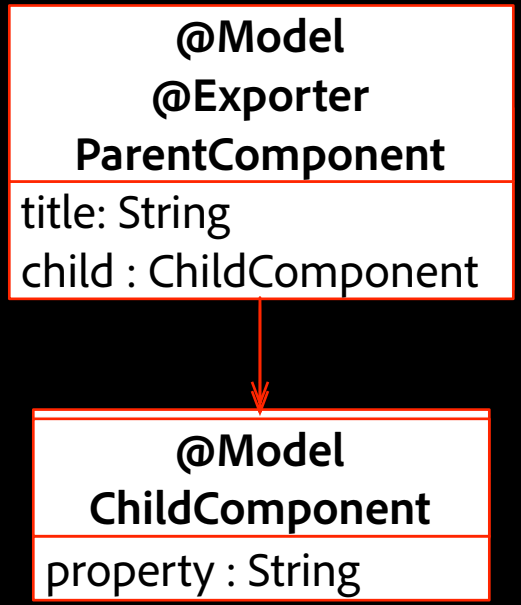
3. The resolved Sling Exporter Servlet invokes the Sling Model Exporter against the Sling Model object adapted from the request or resource (as determined by the Sling Models *adaptables*)

4. The exporter serializes the Sling Model based on the Exporter Options and Exporter-specific Sling Model annotations and returns the result to the Sling Exporter Servlet as a String.

5. The Sling Exporter Servlet returns the String export of the Sling Model as the HTTP Response.

Handling Object Graphs

- Q: Which class needs @Exporter?
- A: Only ParentComponent
- @Exporter goes on the HTTP entrypoint



Other Conversions – Not just for Strings!

- ModelAdapterFactory.exportModel()
- Examples:
  - Jackson – export to Map
  - JAXB – export to DOM Document

adaptTo()

Multiple Exporters – Use @Exporters

```
@Model(adaptables=Resource.class,
    resourceType="myco/something")
@Exporters({
  @Exporter(name = "jackson",
    extensions = "json"),
  @Exporter(name = "gson", selector = "gson",
    extensions = "json")
})
```

adaptTo()

Exporter Options

- Simple map of name/value pairs.
- Semantics are up to ModelExporter impl.
- Specified with
  - exportModel() API
  - @ExporterOption annotation
  - Selectors
  - Request parameters

Exporter Options - Jackson

- “tidy”
- SerializationFeature.*
  - http://bit.ly/jack-serial
- MapperFeature.*
  - http://bit.ly/jack-mapper

ExporterOption Examples

```
@ExporterOption(
    name =
    "SerializationFeature.WRITE_DATES_AS_TIMESTAMPS",
    value = ”false")
```

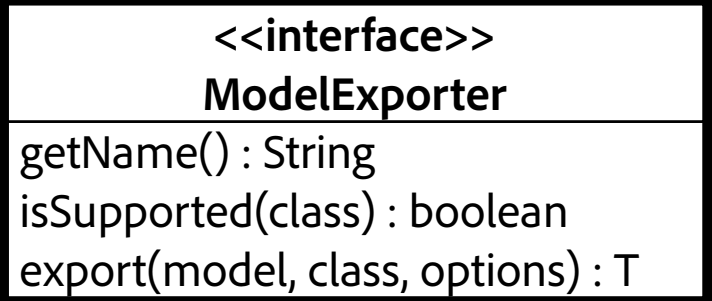
GET /content/something.model.tidy.json

GET /content/something.model.json?tidy=true

Extension Points

Extension Point - ModelExporter

- Defines a new named exporter



ModelExporter Example - GSON

```
public <T> T export(Object model, Class<T> clazz,
      Map<String, String> options)
      throws ExportException {
  return (T) new Gson().toJson(model);
}
```

ModelExporter Example - JAXB

```
public <T> T export(Object model, Class<T> clazz,
      Map<String, String> options) throws ExportException {
  try {
    JAXBContext jaxbContext =
      JAXBContext.newInstance(model.getClass());
    Marshaller marshaller = jaxbContext.createMarshaller();
    StringWriter sw = new StringWriter();
    marshaller.marshal(model, sw);
    return (T) sw.toString();
  } catch (JAXBException e) { throw new ExportException(e); }
}
```

Extension Point – Jackson ModuleProvider

- Customize serialization of classes outside your codebase.
- See http://bit.ly/jack-modules
- Provided Examples:
  - Sling Resources
  - (Http|SlingHttp)?ServletRequest
  - java.util.Enumeration

Jackson ModuleProvider – Joda

```
import org.joda.time.DateTime;

public class TestModel {
   public DateTime getTimestamp() {
      return new DateTime();
   }
}
```

Jackson ModuleProvider – Joda (no Module)

```
"timestamp" : {
  "era" : 1, "dayOfYear" : 242, "dayOfWeek" : 3,
  "dayOfMonth" : 30, "year" : 2017, "weekOfWeekyear" : 35,
  "hourOfDay" : 12, "minuteOfHour" : 33, "monthOfYear" : 8,
  "millisOfDay" : 45184141, "secondOfMinute" : 4, "millisOfSecond" : 141,
  "weekyear" : 2017, "yearOfEra" : 2017, "yearOfCentury" : 17,
  "centuryOfEra" : 20, "secondOfDay" : 45184, "minuteOfDay" : 753,
  "zone" : {
    "fixed" : false,
    "uncachedZone" : {
      "cachable" : true, "fixed" : false, "id" : "America/New_York”
    },
    "id" : "America/New_York”
  },
  "millis" : 1504110784141,
  "chronology" : {
    "zone" : {
      "fixed" : false,
      "uncachedZone" : {
        "cachable" : true  "fixed" : false, "id" : "America/New_York”
      },
      "id" : "America/New_York”
    }
  },
  "afterNow" : false, "beforeNow" : true, "equalNow" : false
}
```

Jackson ModuleProvider - Joda

```
@Component
public class JodaModuleProvider
    implements ModuleProvider {

  public Module getModule() {
    return new JodaModule();
  }
}
```

Jackson ModuleProvider – Joda (w/ Module)

```
"timestamp" : 1504111214012
```

Summary

- Model Objects -> JSON Servlets
  - Just add annotations
  - Supported by Jackson
  - Use ModuleProvider SPI for unowned classes
- Other formats through ModelExporter SPI

Q&A

Thanks!