Lifting the greatness of EMF into the cloud with EMF.cloud

Philip Langer  
planger@eclipsesource.com

Martin Fleck  
mfleck@eclipsesource.com
Building domain-specific (modeling) tools for various domains
Building domain-specific (modeling) tools for various domains
Building domain-specific (modeling) tools for various domains
Building domain-specific (modeling) tools for various domains
EMF.cloud: Enable the adoption of EMF in cloud-based tools

... with less effort, more standard components, and best practices
EMF.cloud

- Common home for components
  - Facilitating or simplifying the usage of EMF & ecosystem
  - Incubate towards becoming standard components
  - Encapsulate best practices

- Scope
  - Middleware for making EMF models accessible to clients
  - Adapters for Ecore and EMF-models (e.g. JSON)
  - Editing support for EMF-based models on the client
  - Tooling for creating Ecore models on the client
  - Example applications that encode best practices
  - ...

© 2019 EclipseSource | http://eclipsesource.com | Philip Langer & Martin Fleck | Lifting the greatness of EMF into the cloud with EMF.cloud
Kick-starting EMF.cloud

1. Coffee IDE: An example modeling tool
2. EMF Model Server
3. Ecore tooling for Theia
Coffee IDE: A Cloud Modeling Tool Example

● Challenges
  ○ Which EMF components can be reused as is on the server?
  ○ Where to put certain custom functionality (client/server)?
  ○ How to integrate them into the client?
  ○ How to set up client-server communication?
  ○ ...

● Example application
  ○ Resembles typical modeling tool
  ○ Blueprint for building custom modeling tools
  ○ Driving development of generic components
  ○ Best practices
Coffee IDE: A Cloud Modeling Tool Example

Building web-based tools with Theia and Che
Tuesday, 16:15 - 16:50, Theater Stage
EMF Model Server

● EMF is perfectly suitable on the server
  ○ With or without OSGi
  ○ Loading, model manipulation, and serialisation models
  ○ Existing language implementations work out of the box

● Why is a new model server needed?
Why we needed a model server
Why we needed a model server
Why we needed a model server
Why we needed a model server
Why we needed a model server
Why we needed a model server
EMF Model Server

- **Runtime state of loaded models (“shared editing domain”)**
  - Share potentially “dirty” runtime state of a model
  - Command-based change interface
  - Notification mechanism via sockets

- **Java-based server**
  - Reuse existing model implementations
  - Based on popular REST framework

- **Convenient model access**
  - REST API: create, update, patch, save, ... models
  - Multiformat: JSON or XMI supported
  - Multiplatform: Java and JavaScript-based client API
EMF Model Server

- Initialize model server for workspace folder
- Load or get model

```java
ModelServerClient client = new ModelServerClient("http://localhost:8081/api/v1/");
client.get("foo.xmi")
  .thenAccept(response -> ... do something with response.body() ...);
```

`foo.xmi` is only loaded once per session!

Other methods: `getSchema()`, `delete()`, `update()`, `save()`, ...
Subscribe, react to changes, optionally synchronize

```java
client.subscribe("foo.xmi&format=json", new JsonToEObjectSubscriptionListener() {

    private final CommandCodec codec = new DefaultCommandCodec();

    @Override
    public void onIncrementalUpdate(EObject message) {
        Command command = codec.decode(editingDomain, (CCommand) message);
        editingDomain.getCommandStack().execute(command);
    }

});
```

Subscription via sockets and serialized commands!
EMF Model Server

Patch models via commands

```json
{
  "type": "add",
  "owner": {
    "eClass": "http://www.eclipsesource.com/example/coffeemodel#//AutomaticTask",
    "$ref": "SuperBrewer3000.xmi//@workflows.0"
  },
  "feature": "nodes",
  "objectValues": [
    {
      "eClass": "http://www.eclipsesource.com/example/coffeemodel#//AutomaticTask",
      "$ref": "//@commands.1/@objectsToAdd.0"
    }
  ],
  "objectsToAdd": [
    {
      "eClass": "http://www.eclipsesource.com/example/coffeemodel#//AutomaticTask",
      "name": "Brew"
    }
  ],
  "indices": [ 1 ]
}
```
EMF Model Server

Patch models via commands

- Model server applies command
  - If not, reject and let client know
  - If applicable, confirms execution to sender
EMF Model Server

Patch models via commands

- Model server applies command
  - If not, reject and let client know
  - If applicable, confirms execution to sender
- Model server notifies listeners
- Listeners can synchronize (or react how they like)

```java
client.subscribe("foo.xmi\&format=json", new JsonToObjectSubscriptionListener() {
    private final CommandCodec codec = new DefaultCommandCodec();
    @Override
    public void onIncrementalUpdate(EObject message) {
        Command command = codec.decode(editingDomain, (CCommand) message);
        editingDomain.getCommandStack().execute(command);
    }
});
```
EMF Model Server: Demo
Ecore tooling for Theia

- Ecore Tooling
  - Ecore diagram editor in Theia
  - Based on GLSP + EMF Model Server
  - 100 % XMI compatibility
    → code generation, validation, etc.

- Soon
  - Brushing up rendering and UI
  - Ecore property view
  - Ecore tree model explorer
Summary

- **Eclipse EMF.Cloud**
  https://projects.eclipse.org/projects/ecd.emfcloud
  - EMF Model Server → IP review
  - Coffee IDE → in progress
  - Ecore Tooling → in progress

- **More to come**
  - Theia EMF UI components, ...
  - Your contribution?

- **Our common home for EMF in the Cloud!**
  - planger@eclipsesource.com