Use strongly-typed models

November 23, 2022 • Jan Cerman • 3 min read • Java

Strongly-typed models are programmatic representations of the content types in your Kontent.ai project. They help you map the JSON objects you get from Delivery REST API directly to a model you can use in your code.

The mapping works a little differently for each platform. We recommend that you check the docs of the SDK you're using:

- Using strongly typed models with Delivery .NET SDK
- Using strongly typed models with Delivery JavaScript/TypeScript SDK
- Using strongly typed models with Delivery Java SDK
- Using strongly typed models with Delivery PHP SDK

Life without strongly typed models

When getting content directly from the Delivery REST API, you receive content items as JSON objects. Your app then needs to parse the JSON to display your content. For example, to access the *Headline* element in a content item, you'd use a notation like this `response.data.item.elements.headline.value`.

In a more complicated scenario, this approach becomes cumbersome. It forces you to remember the JSON structure of your content items and the codenames of your content elements.

To make this easier, use a Delivery SDK to map the retrieved content items to their **strongly typed models**.

Use strongly typed models

This practice has several advantages:

- type safety during compile-time
- convenience (IntelliSense remembers content type properties for you)
- support of type-dependent functionalities

The models are plain classes that don't have any attached behavior or dependency on an external framework. Each model corresponds to a content type in your Kontent.ai project.

To create a content type's representation in code, you need a class with properties representing the individual content elements:

- Each property is mapped to its content element's codename either explicitly or using a naming convention. For example, codename `body_text` becomes property `bodyText`.
- Properties are usually typed using classes provided by the SDK.

- Depending on the SDK, simple content elements (like text and number) are sometimes represented using native types (like string and integer).

The following example is a strongly typed model of the *Homepage* content type.

Java

```java
// Tip: Find more about Java SDK at https://kontent.ai/learn/java
import kontent.ai.delivery.Asset;
import kontent.ai.delivery.ContentItemMapping;
import kontent.ai.delivery.ElementMapping;
import kontent.ai.delivery.System;
import java.lang.String;
import java.util.List;

/**
 * This code was generated by a <a href="https://github.com/kontent-ai/java-packages/tree/master/delivery-sdk-generators">kontent-generators-java tool</a>
 *
 * Changes to this file may cause incorrect behavior and will be lost if the code is regenerated.
 * For further modifications of the class, create a separate file and extend this class.
 */
@ContentItemMapping("homepage")
public class Homepage {
    @ElementMapping("body_text")
    String bodyText;

    @ElementMapping("headline")
    String headline;

    @ElementMapping("picture")
    List<Asset> picture;

    System system;

    public String getBodyText() {
        return bodyText;
    }

    public void setBodyText(String bodyText) {
        this.bodyText = bodyText;
    }

    public String getHeadline() {
        return headline;
    }

    public void setHeadline(String headline) {
        this.headline = headline;
```

```
    }

    public List<Asset> getPicture() {
        return picture;
    }

    public void setPicture(List<Asset> picture) {
        this.picture = picture;
    }

    public System getSystem() {
        return system;
    }

    public void setSystem(System system) {
        this.system = system;
    }
}
```

Generate models

We recommend that you generate the models using the Java model generator.

Use the generator to create strongly typed models from your Kontent.ai project by providing your Project ID.

Java

```
// Find instructions on using the Java model generator at https://github.com/kontent-ai/java-packages/tree/master/delivery-sdk-generators
import com.squareup.javapoet.JavaFile
import kontent.ai.delivery.DeliveryClient
import kontent.ai.delivery.DeliveryOptions
import kontent.ai.delivery.generators.CodeGenerator

buildscript {
    repositories {
        mavenCentral()
    }
    dependencies {
        classpath('ai.kontent:kontent-delivery-generators:latest.release')
    }
}

// showcase task
task generateModels {
    doLast {

        // The most complex solution, you could configure the client as you want
        // i.e. set preview API key
```

```
        DeliveryOptions options = new DeliveryOptions();
        options.setProjectId("975bf280-fd91-488c-994c-2f04416e5ee3");
        DeliveryClient client = new DeliveryClient(options);

        CodeGenerator generator = new CodeGenerator(
            options.getProjectId(),
            'ai.kontent.test.springapp.models',
            file('src/main/java')
        );
        List<JavaFile> sources = generator.generateSources(client);
        generator.writeSources(sources);
    }
}
```

Get strongly typed content

With your models defined and added to your application, you can use them when retrieving items from Kontent.ai.

Java

```
// Tip: Find more about Java SDK at https://kontent.ai/learn/java
import kontent.ai.delivery.*;

// Initializes a DeliveryClient
DeliveryClient client = new DeliveryClient("8d20758c-d74c-4f59-ae04-ee928c0816b7");

// Create strongly typed models according to https://kontent.ai/learn/strongly-typed-models
// Registers the model class for navigation items
client.registerType(Homepage.class);

CompletionStage<Homepage> homepageResult = client.getItem("hello_caas_world",
Homepage.class);
// Use homepageResult
// homepageResult.thenAccept(homepage -> System.out.println(homepage.getHeadline())
```

What's next?

You'll often have to define how your application resolves rich text elements containing assets, links, components, and other content items. For more advanced examples of using strongly typed models, see our sample applications.