Designing Connected Content

Final Presentation

Group 2

26.01.2022

Santiago Juste Valverde, Jaime Martinez de Miguel, Michaela Meindl

Agenda

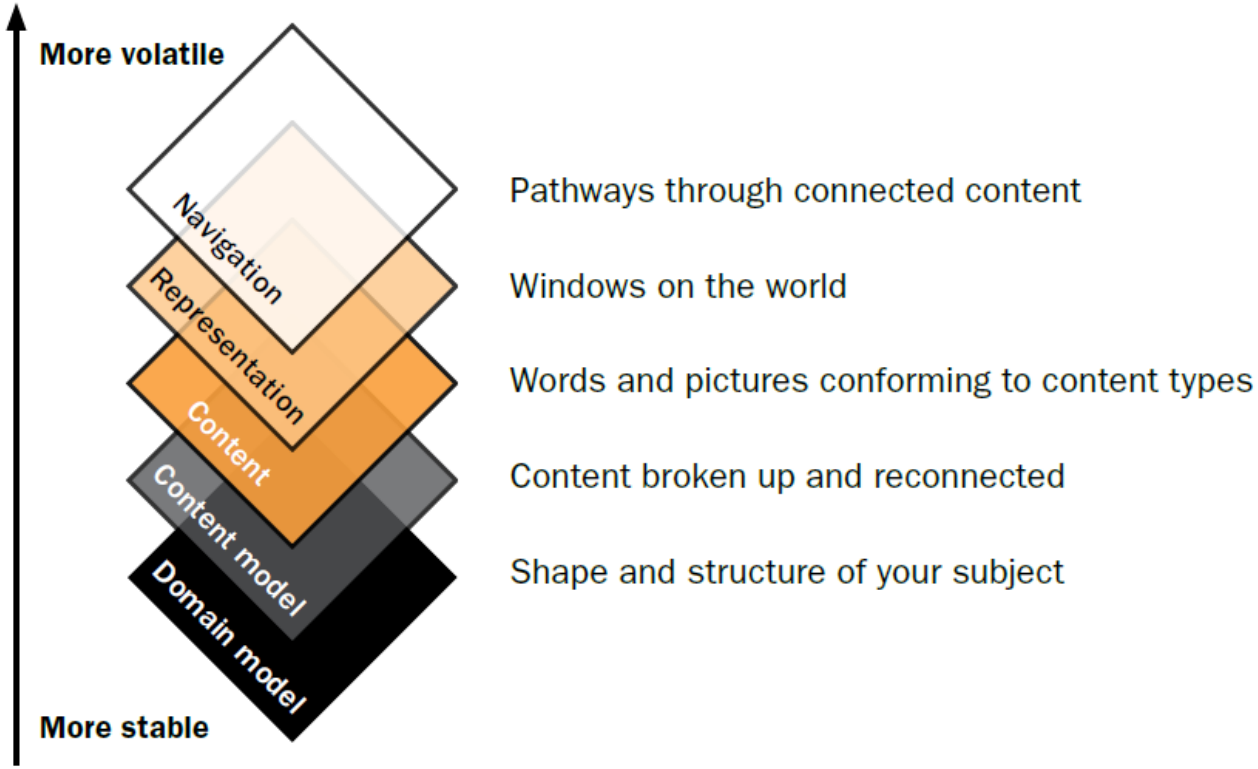
1. Introduction
2. Domain Model
3. Content Model
4. Content Management System (Contentful)
5. Static versus Dynamic Websites
6. Application of Concepts in Example Conference Website

Introduction

- Structured way of modeling a domain.
- Prevention of “loss” of content.
- Putting content into context.
- Separation of content and structure from presentation.
- Providing a possibility to publish content to people who don’t have special technical knowledge.
- Based on Atherton, Mike, and Carrie Hane (2017): Designing Connected Content.

Atherton, Mike, and Carrie Hane (2017): *Designing Connected Content*.

The Structured Content Stack



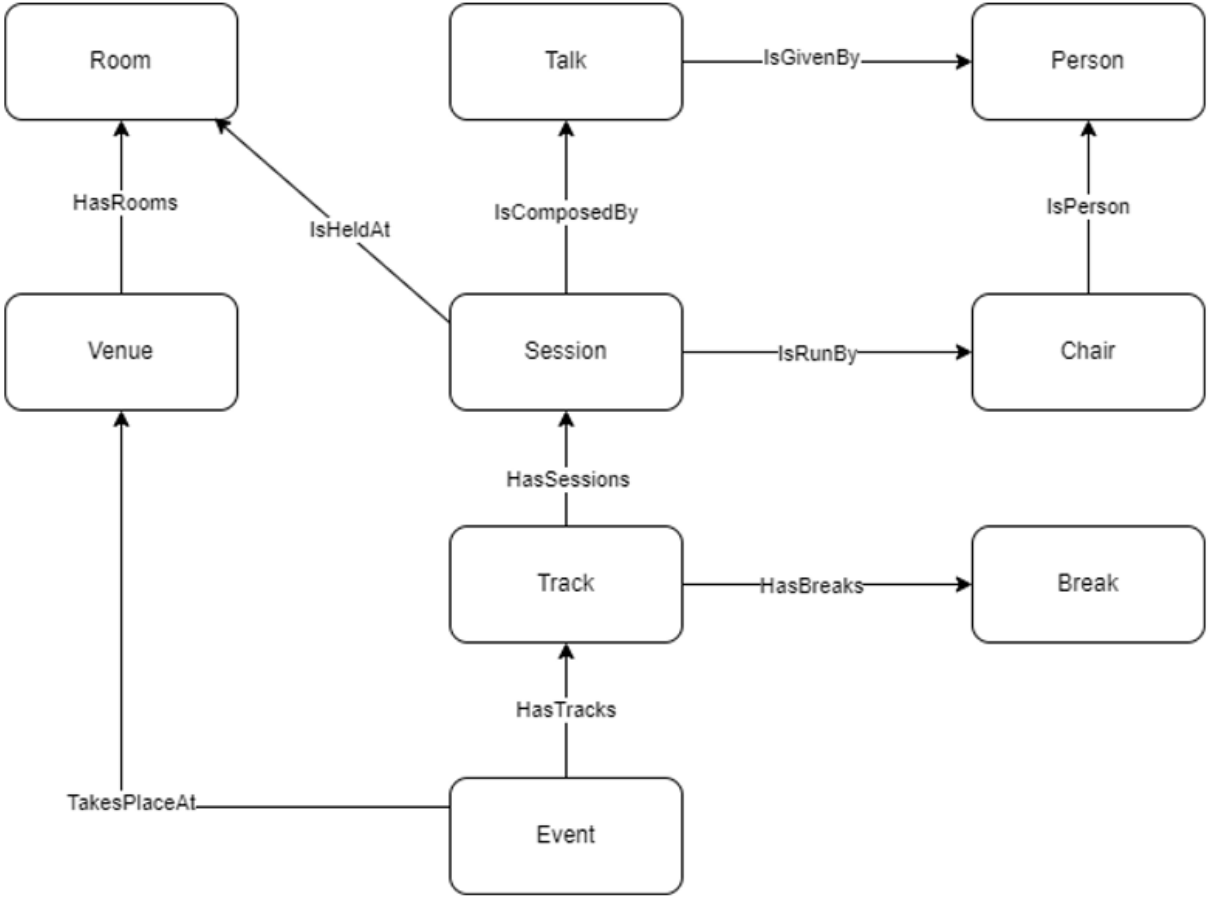
Atherton, Mike, and Carrie Hane (2017): *Designing Connected Content*. p. 169

Domain Model

Domain Model

- Conceptual model of the subject domain.
- Shows concepts and their relationships.
- Makes sure the content addresses user needs.
- Gives content structure.

Atherton, Mike, and Carrie Hane (2017): *Designing Connected Content*.

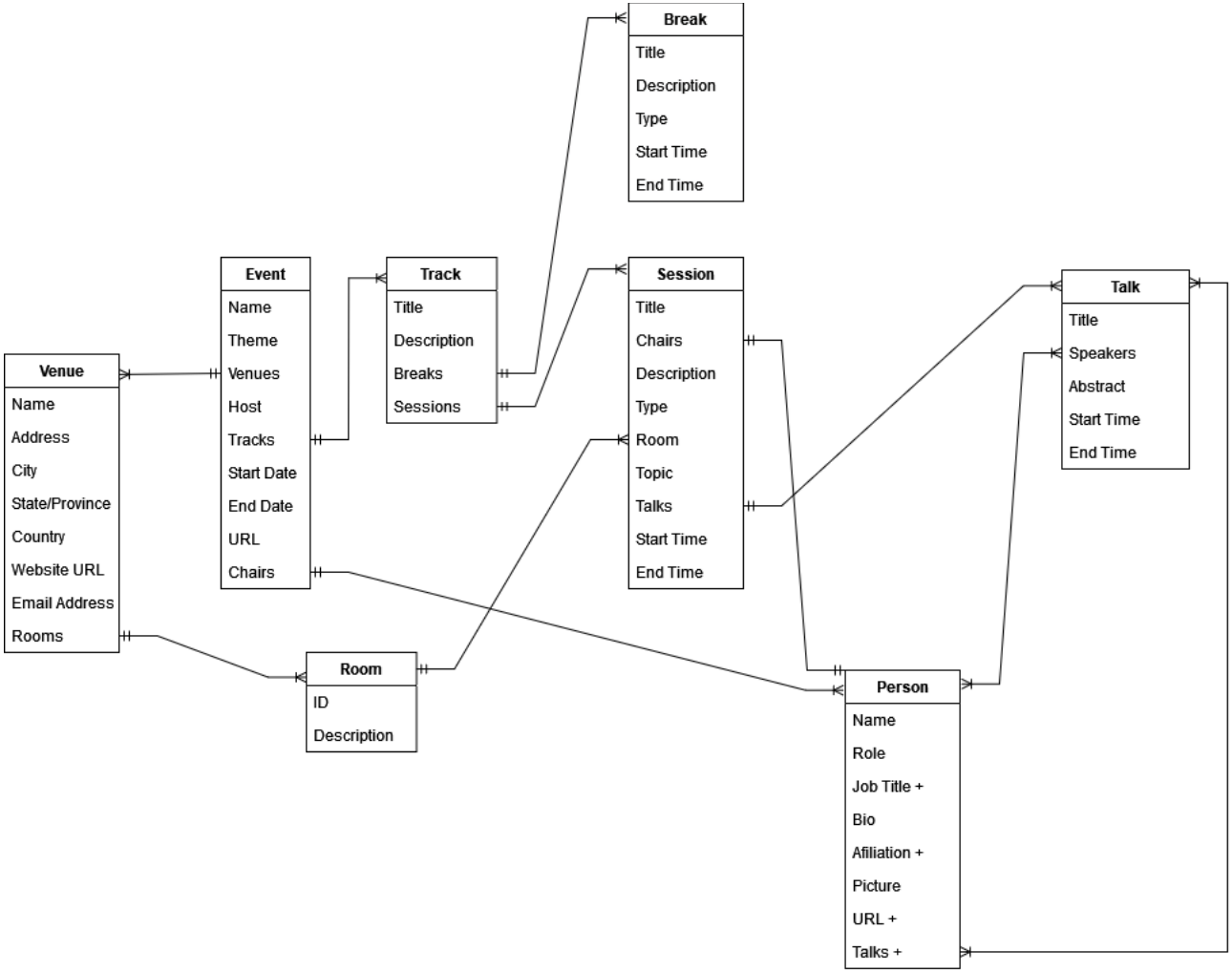

Room
Talk
IsGivenBy
Person
HasRooms
IsHeldAt
IsComposedBy
IsPerson
Venue
Session
IsRunBy
Chair
HasSessions
Track
HasBreaks
Break
HasTracks
TakesPlaceAt
Event

Content Model

Content Model

- Unnecessary objects removed.
- Multiple objects can be combined into one content type.
- Content types define attributes (= characteristic or quality of an object).
- Representation considered for decisions on model.
- Requires a lot of detailed thinking.
- Reconnection of content types.

Atherton, Mike, and Carrie Hane (2017): *Designing Connected Content*.


Break
Title
Description
Type
Start Time
End Time
Event
Name
Theme
Venues
Host
Tracks
Start Date
End Date
URL
Chairs
Track
Title
Description
Breaks
Sessions
Session
Title
Chairs
Description
Type
Room
Topic
Talks
Start Time
End Time
Talk
Title
Speakers
Abstract
Start Time
End Time
Venue
Name
Address
City
State/Province
Country
Website URL
Email Address
Rooms
Room
ID
Description
Person
Name
Role
Job Title +
Bio
Afiliation +
Picture
URL +
Talks +

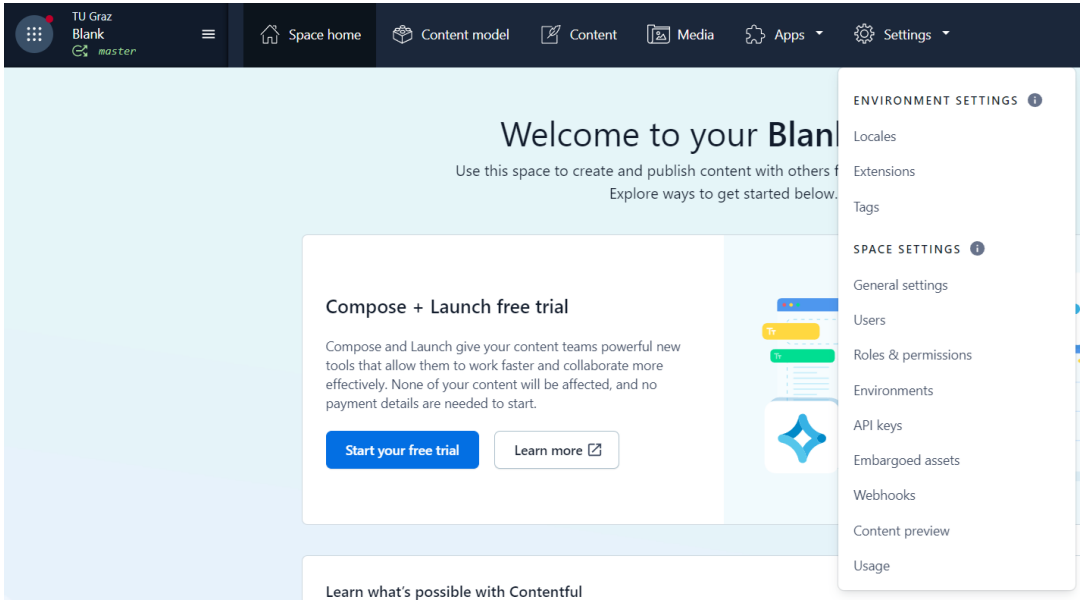
Content Management System

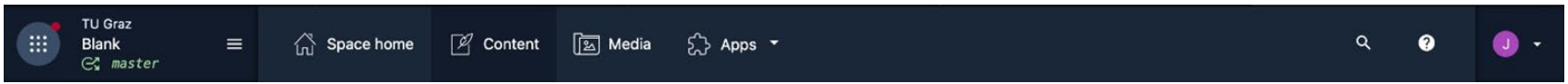
Content Management Systems

- Software used to manage the creation and modification of digital content.
- 2 major components:
  - Content management application (CMA): frontend UI for adding, modifying and removing content.
  - Content delivery application (CDA): backend, which computes the content and updates the website via an API.
- Typical features are:
  - Indexing
  - Search
  - Retrieval
  - Format management
  - Revision control and management

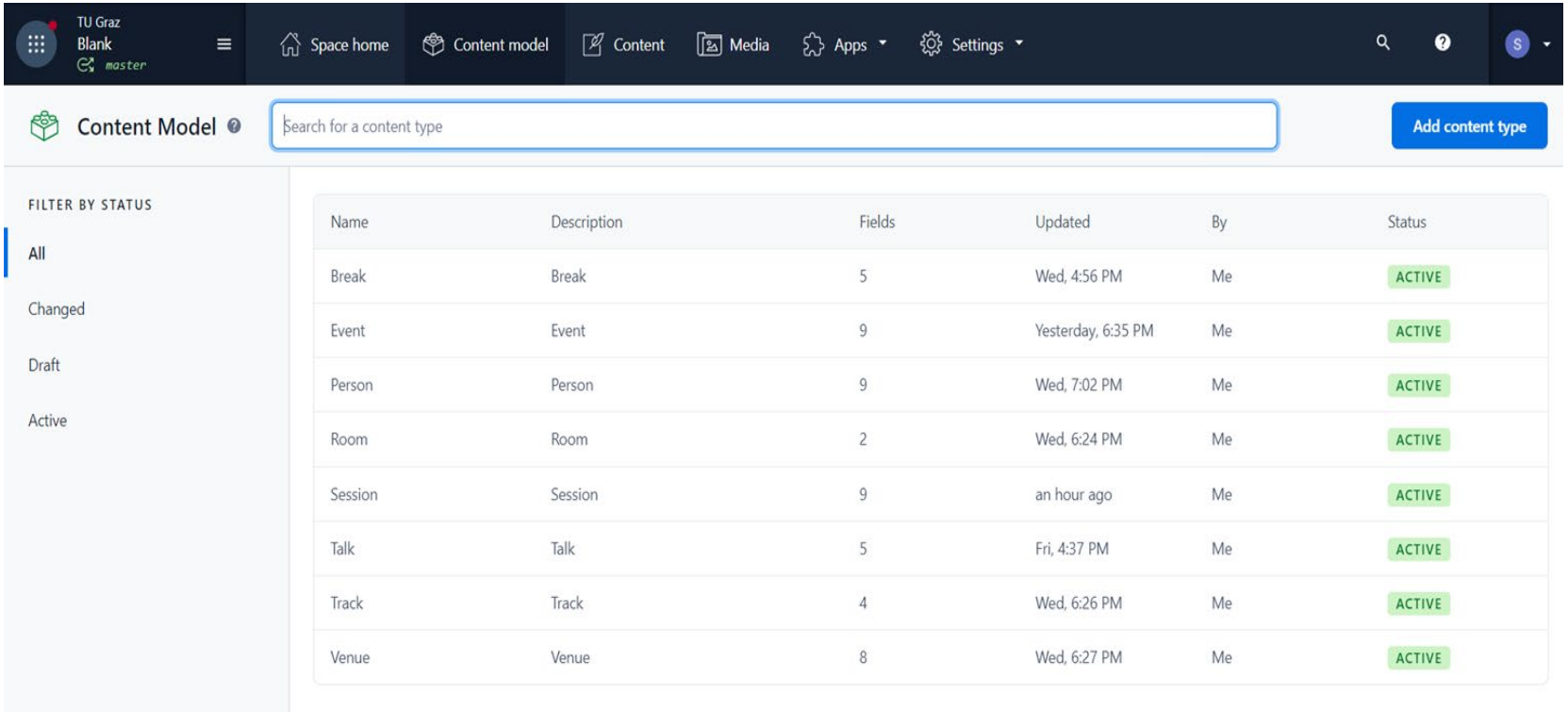
https://kinsta.com/knowledgebase/content-management-system/

Contentful

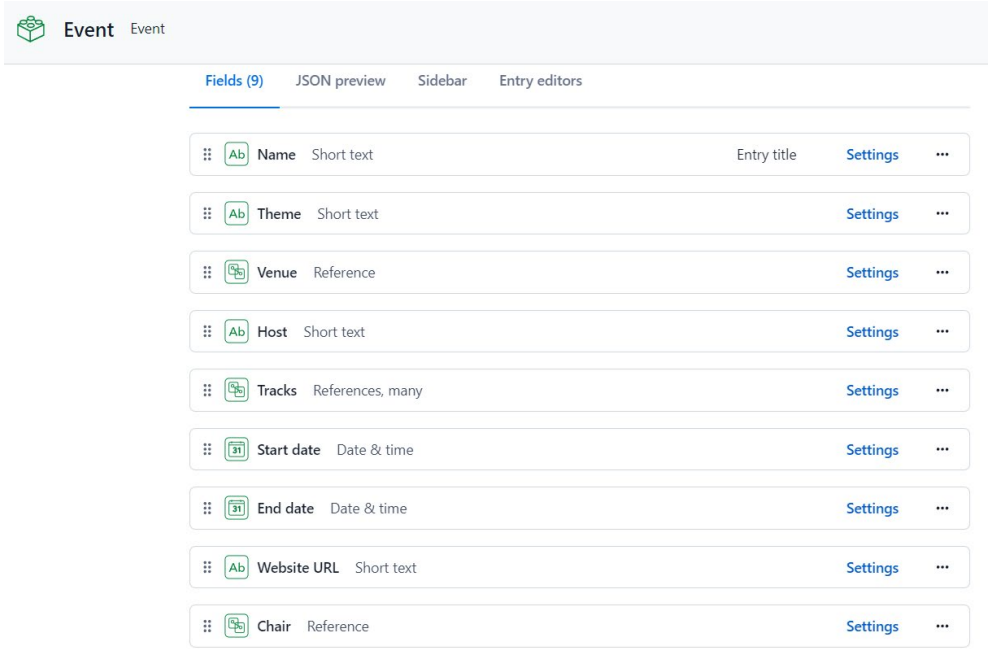

TU Graz
Blank
master
Space home
Content model
Content
Media
Apps
Settings
ENVIRONMENT SETTINGS
Locales
Extensions
Tags
SPACE SETTINGS
General settings
Users
Roles & permissions
Environments
API keys
Embargoed assets
Webhooks
Content preview
Usage
Welcome to your Blank
Use this space to create and publish content with others
Explore ways to get started below.
Compose + Launch free trial
Compose and Launch give your content teams powerful new tools that allow them to work faster and collaborate more effectively. None of your content will be affected, and no payment details are needed to start.
Start your free trial
Learn more
Learn what's possible with Contentful

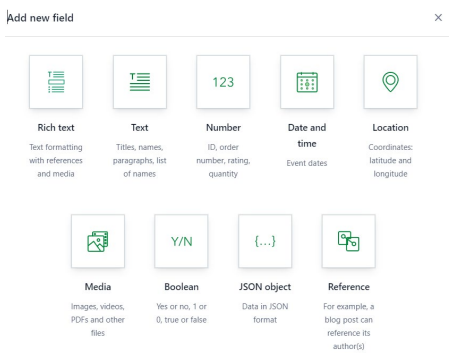

TU Graz
Blank
master
Space home
Content
Media
Apps
J

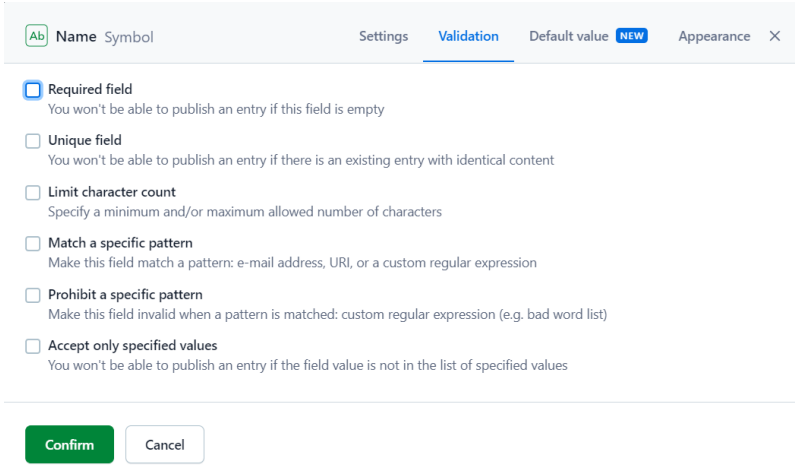
Contentful - Content Model (1/3)



Contentful - Content model (2/3)







Contentful - Content Model - JSON (3/3)

Fields (9) | JSON preview | Sidebar | Entry editors


```
{
  "name": "Event",
  "description": "Event",
  "displayField": "name",
  "fields": [
    {
      "id": "name",
      "name": "Name",
      "type": "Symbol",
      "localized": false,
      "required": true,
      "validations": [],
      "disabled": false,
      "omitted": false
    },
    {
      "id": "theme",
      "name": "Theme",
      "type": "Symbol",
      "localized": false,
      "required": true,
      "validations": [],
      "disabled": false,
      "omitted": false
    },
    {
      "id": "venue",
      "name": "Venue",
      "type": "Link",
      "localized": false,
      "required": true,
      "validations": [],
      "disabled": false,
      "omitted": false,
      "linkType": "Entry"
    },
    {
      "id": "tracks",
      "name": "Tracks",
      "type": "Array",
      "localized": false,
      "required": true,
      "validations": [],
      "disabled": false,
      "omitted": false,
      "items": {
        "type": "Link",
        "validations": [],
        "linkType": "Entry"
      }
    },
    {
      "id": "startDate",
      "name": "Start date",
      "type": "Date",
      "localized": false,
      "required": true,
      "validations": [],
      "disabled": false,
      "omitted": false
    },
    {
      "id": "endDate",
      "name": "End date",
      "type": "Date",
      "localized": false,
      "required": true,
      "validations": [],
      "disabled": false,
      "omitted": false
    },
    {
      "id": "websiteUrl",
      "name": "Website URL",
      "type": "Symbol",
      "localized": false,
      "required": true,
      "validations": [
        {
          "regexp": {
            "pattern": "^(ftp|http|https):\\/\\/(\\w+:{0,1}\\w*@)?(\\S+)(:[0-9]+)?(\\/|\\
/([\\w#!:.?+=&%@!\\-/]))?$",
            "flags": null
          }
        }
      ],
      "disabled": false,
      "omitted": false
    },
    {
      "id": "chair",
      "name": "Chair",
      "type": "Link",
      "localized": false,
      "required": true,
      "validations": [],
      "disabled": false,
      "omitted": false,
      "linkType": "Entry"
    }
  ],
  "sys": {
    "space": {
      "sys": {
        "type": "Link",
        "linkType": "Space",
        "id": "x4pzlsnfjt0t"
      }
    },
    "id": "event",
    "type": "ContentType",
    "createdAt": "2022-01-11T13:04:46.292Z",
    "updatedAt": "2022-01-17T17:35:23.664Z",
    "environment": {
      "sys": {
        "id": "master",
        "type": "Link",
        "linkType": "Environment"
      }
    },
    "publishedVersion": 25,
    "publishedAt": "2022-01-17T17:35:23.664Z",
    "firstPublishedAt": "2022-01-11T13:04:46.879Z",
    "createdBy": {
```

Static versus Dynamic Websites

Static Websites

- Web pages are returned by the server as prebuilt source code files
- No processing of content on the server
- Fast, as no changes are done on the server
- Cannot be manipulated on server or interact with databases.
- Example for static site generator: Eleventy

https://www.geeksforgeeks.org/static-vs-dynamic-website/

Static Site Generator: Eleventy

- Uses independent template engines.
- Content decoupled from host.
- Works with multiple template languages like HTML, Markdown, JavaScript, Nunjucks, Liquid etc.
- Works with any project directory structure.

https://www.11ty.dev/docs/

Dynamic Websites

- Web pages are built during runtime.
- Deploying web pages with help of scripting languages such as PHP, Node.js, ASP.NET etc.
- Updates and interaction with databases are possible
- More commonly used as updates can be done very easily

https://www.geeksforgeeks.org/static-vs-dynamic-website/

Conference Website

Eleventy Configuration File: *eleventy.js*

```
const { documentToHtmlString } = require("@contentful/rich-text-html-renderer");
const eleventyNavigationPlugin = require("@11ty/eleventy-navigation");
const dateFilter = require("nunjucks-date-filter");

module.exports = function (eleventyConfig) {
  eleventyConfig.addPassthroughCopy("./src/css");
  eleventyConfig.addWatchTarget("./src/css/");
  eleventyConfig.addShortcode("pathPrefix", () => process.env.MY_PREFIX);
  eleventyConfig.addPlugin(eleventyNavigationPlugin);
  eleventyConfig.addNunjucksFilter("date", dateFilter);
  eleventyConfig.addFilter("renderRichTextAsHtml", (value) =>
      documentToHtmlString(value)
  );
  return {
    dir: {
      input: "src",
      output: "public",
    },
  };
};
```

Utilities File for Contentful-Client: *.env* and *utils.js*

Creating .env file to store host-, space-, and access-keys.

Subsequently, they are processed in utils.js file

```
require("dotenv").config();

const contentful = require("contentful");
const contentfulClient  = contentful.createClient({
  host: process.env.CTFL_HOST,
  space: process.env.CTFL_SPACE,
  accessToken: process.env.CTFL_ACCESSTOKEN,
});

exports.contentfulClient = contentfulClient;
```

Get Entries of Content Type: *persons.js*

```
const client = require('../utils').contentfulClient;

module.exports = async () => {
  // create a request for all entries that match person type.
  const persons = await client.getEntries({
    content_type: 'person',
  });
  return persons.items;
};
```

Get Assets: *images.js*

```
const client = require('../utils').contentfulClient;
const Image = require("@11ty/eleventy-img");

module.exports = async () => {
  await client
    .getAssets()
    .then(function (assets) {
        assets.items.map(function (asset) {
            (async () => {
                var imageURL = 'https:' + asset.fields.file.url;
                let stats = await Image(imageURL, {
                    widths: [200],
                    formats: ["jpeg"],
                    outputDir: "./public/conf/img/",
                    filenameFormat: function (id, src, width, format, options) {
                        const personName = asset.fields.title;
                        const name = personName.replace(" ", "");
                        return `${name}.${format}`;
                    }
                });
            })()
            .catch(function (e) {
            console.log(e);
            });
        });
    });
};
```

Base-Layout File: *layout.njk*

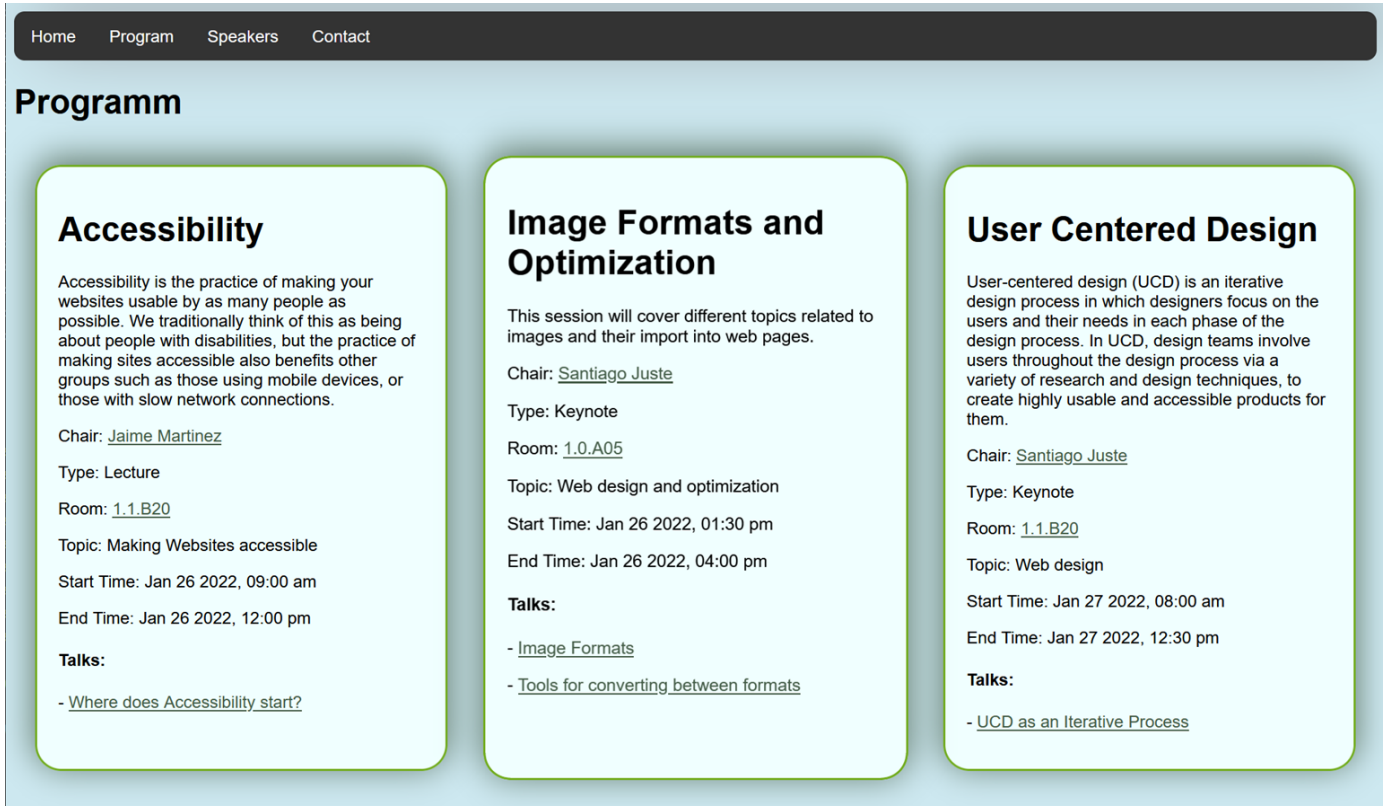
```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>{{ title }}</title>
        <link rel="stylesheet" href="../../css/style.css" />
    </head>
  <header>
{% set navPages = collections.all | eleventyNavigation %}
<ul>
{%- for entry in navPages %}
  <li{% if entry.url == page.url %} class="my-active-class"{% endif %}>
    <a href="../..{{ entry.url | url }}">{{ entry.title }}</a>
  </li>
{%- endfor %}
</ul>
  </header>
    <body>
        {{ content | safe }}
    </body>
</html>
```

Example Page File: *persons.njk*

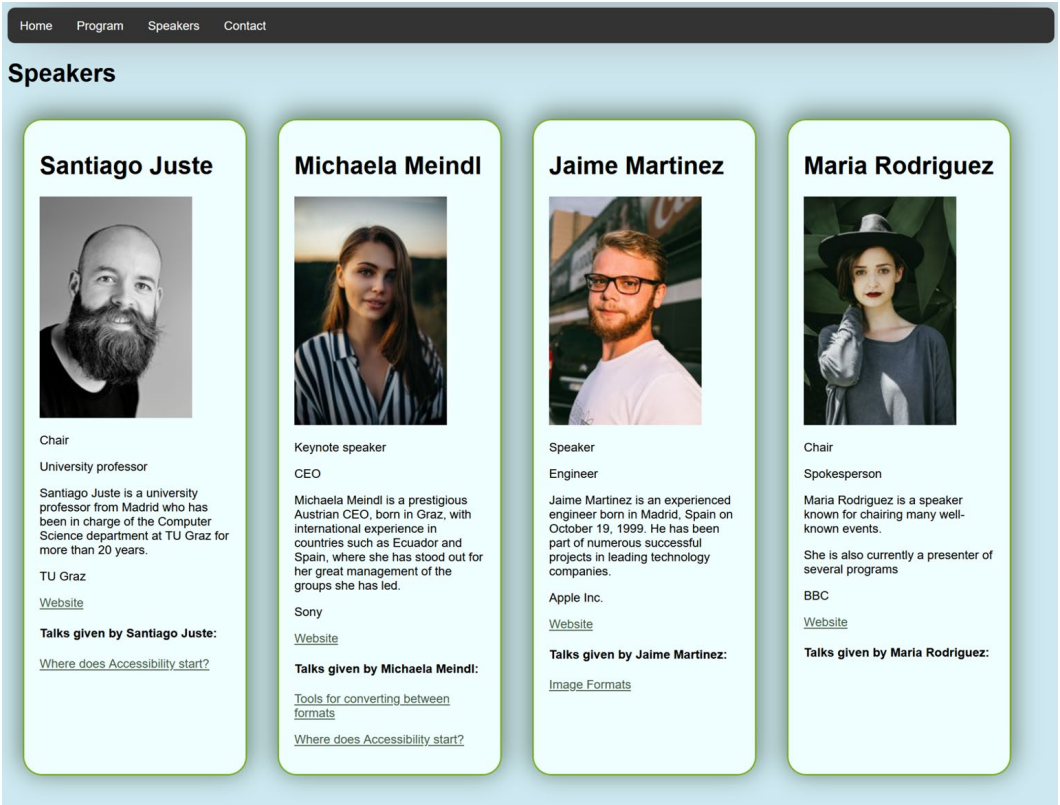
```
---
layout: layout.njk
pagination:
    data: persons
    size: 1
    alias: person
    tags: person
    addAllPagesToCollections: true
permalink:  "./conf/{{ person.fields.name | slug }}/"
eleventyComputed:
    title: "{{ person.fields.name }}"
---
<main>
    <div class="person">
        <h1>{{ person.fields.name }}</h1>
            {% set parsedUrl %} {% pathPrefix %}img/{{person.fields.name}}.jpeg {%endset%}
            {% set finalUrl %}{{ parsedUrl | replace(" ", "") }} {% endset %}
        <img src={{finalUrl}} />
        <p>Job Title: {{person.fields.jobTitle}}</p>
        <p>Bio: {{ person.fields.biography | renderRichTextAsHtml | safe }}</p>
        <p>Affiliation: {{person.fields.afiliation}}</p>
        <a href= "{{person.fields.url}}", target="_blank">Website</a>
        {% for talk in person.fields.talks %}
            <p> Talks given by {{ person.fields.name }}: <br>
            <a href="{% pathPrefix %}{{talk.fields.topic | slug }}">{{talk.fields.topic}}</a>
            </p>
        {% endfor %}
    </div>
</main>
```

Website Homepage

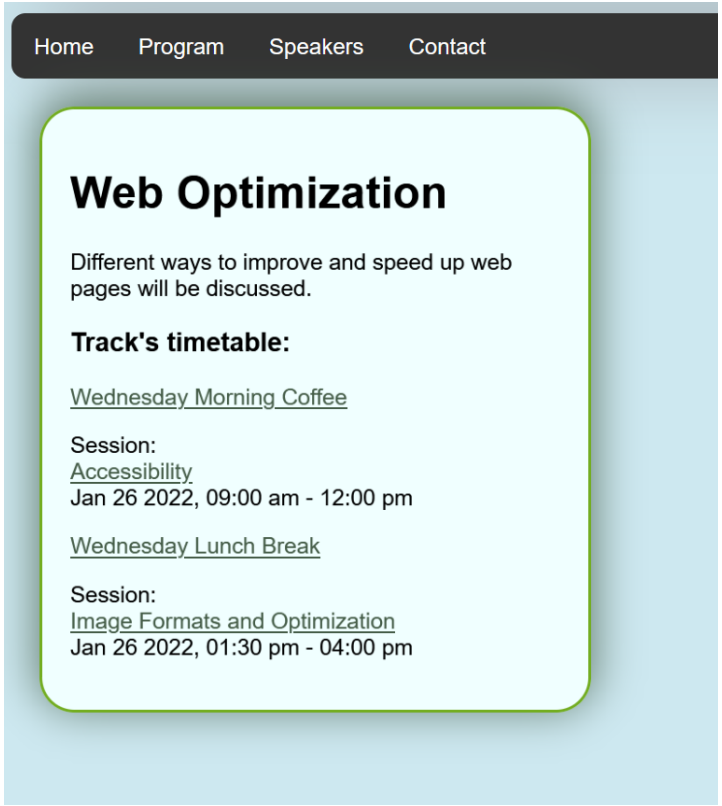
Website Programm Page



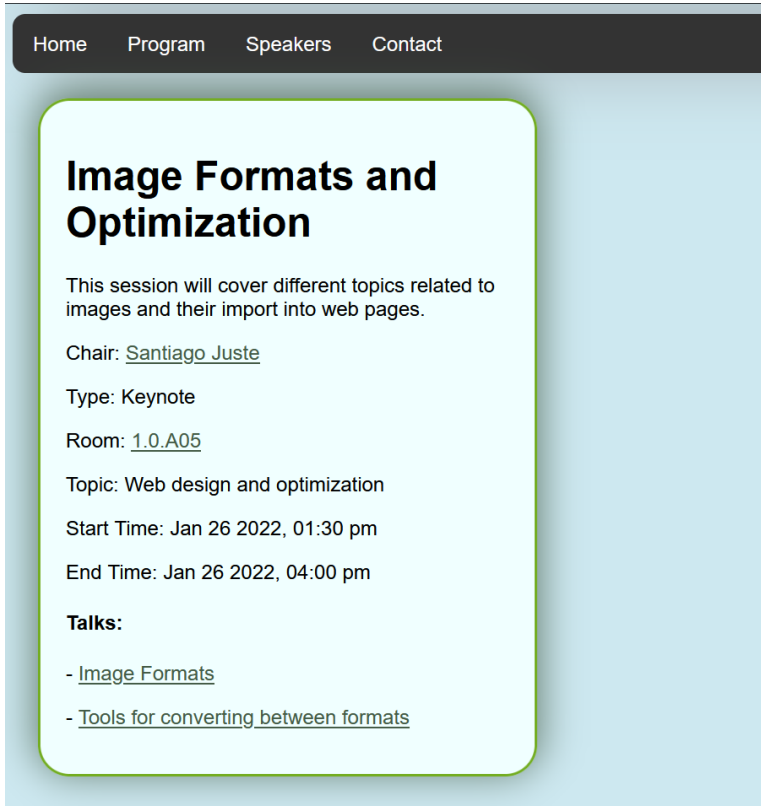
Website Speakers Page



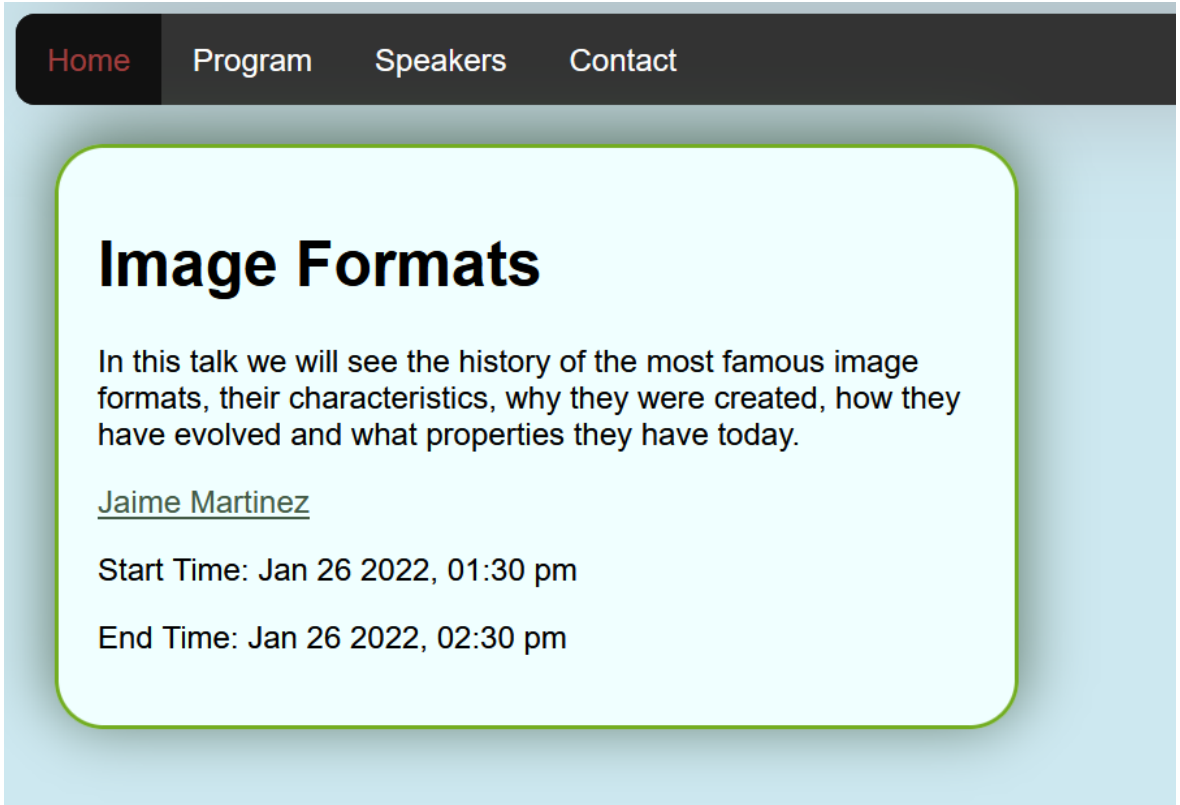
Website Example Track



Website Example Session



Website Example Talk



Conference Website Video

https://youtu.be/hfMtCr3XQ5I

Conclusions

- Designing connected content gives structure to the whole project
- Small pieces of content are easily adaptable and reusable
- Separation of responsibilities in teams - still everybody working on the same domain