Exercise Sheet 1 (SS 2022)

3.0 VU Semistructured Data

Information on the Exercise Sheet

General

In the first exercise sheet you will develop a schema for a file format. First, you will create an XML-Schema (XSD) and write a matching XML document. In the second part, you will convert the XML-Schema into a Document Type Definition (DTD).

Submit your solution as a single ZIP file (max. 5MB). The ZIP file should contain an XML-Schema, a DTD as well as fitting XML documents for the XML-Schema and the DTD. Thus, you should submit the following files:

- `screening-plan.xsd`
- `screening-plan-xsd.xml`
- `screening-plan.dtd`
- `screening-plan-dtd.xml`

The exercise sheet is made up of 5 tasks (described below). You can earn a total of 10 points.

Deadline

at the latest April 28th	**12:00**	Upload your submission on TUWEL
Please do not forget!	$\Longrightarrow$	Register for an exercise interview in TUWEL

Exercise Interviews

During the solution discussion, the correctness of your solution as well as your understanding of the underlying concepts will be assessed.

The scoring of your submission is primarily based on your performance at the solution discussion. Therefore, it is possible (in extreme cases) to receive 0 points although the submitted solution was technically correct.

Please, be punctual to your solution discussion. Otherwise, we cannot guarantee that your full solution can be graded in your assigned time slot. Remember to bring your student id to the solution discussion. It is not possible to assess your solution without an id.

Office Hours (voluntary)

On the 06.04.2022 and 27.04.2022, we will offer in-person office hours for the first exercise sheet. If you have technical issues, trouble understanding the tasks of this sheet, or other questions then please come to the office-hours. We will gladly answer your questions and help you with your problems. Please note that the goal of the office hours is to help you understand the course material and not to solve the exercise sheet for you or to correct your solution of the exercise sheet in advance. Participation is entirely voluntary. Date and place of the office hours will be announced in TUWEL.

We also recommend that you get involved in the forum and actively discuss with your colleagues on the forum. From experience we believe that this helps all parties in the discussion greatly to improve their understanding of the material.

Further Questions – TUWEL Forum

If you have any further questions, regarding the organization or material, you can use the TUWEL forum.

Exercises: XML Schema

First, you will write an XML schema for managing a movie screening plan. Save the created XML-Schema in the `screening-plan.xsd` file. In task 3, you will then be asked to create an XML document `screening-plan-xsd.xml` that matches the schema.

Important: Make sure that your schema file is well-formed and that your XML document is valid! If this is not the case you will receive 0 points for the associated tasks! If you have trouble implementing all aspects of the schema you still have to make sure that your schema is well-formed and the document is valid.

Aufgabe 1 (Defining the Elements of `screening-plan.xsd`) *[4 Punkte]*

The XML-Schema shall validate XML documents with the following structure:

Element `screening-plan`. The root element `screening-plan` stores all the information of the screening plan and contains the following elements in *any order exactly once*:

- An `actors` element;
- a `screenings` element;
- a `movies` element.

Element `actors`. The `actors` element stores all necessary information of movie actors. Therefore, it consists of an arbitrary number of `actor` elements.

Element `actor`. An `actor` element stores all important information about an actor. Therefore, it contains the following elements in the specified order: One `first-name` element containing the first name of the actor. Followed by, at a minimum 0 and at a maximum 2 `middle-name` elements that contain the middle names of the actor. Finally, the content of this element ends with one `last-name` element containing the last name of the actor.

Furthermore, the `actor` element has the following attributes:

- An *optional* `birth_country`, containing the country of birth of the actor.
- an *optional* `birth_date`, containing the date of birth of the actor.
- a *required* `aid` that identifies the actor. It adheres to the following format: An `aid` starts with "Act" followed by at least one digit. For example: "Act2628".

Element `screenings`. This subtree stores all the information of movie screenings in cinemas. To do so, the `screenings` element contains an arbitrary number of `cinema` elements.

Element `cinema`. A `cinema` element contains *at least* one pair of the following elements in the specified order: a *necessary* `movie` element and an *optional* `info` element that holds detailed information about the screening. For instance, a valid example of the content of the `cinema` element would be: "`movie`, `info`, `movie`, `movie`, `info`", where information about the screening of the first and the last movie is available, while information about the second movie is absent.

Element `movie` (Child of `cinema`). The `movie` element is empty and has solely a string attribute `movie_ref`.

Element `info`. An `info` element has an attribute `announcement` that stores the date of the announcement of the movie screening. Furthermore, the `info` element stores information about the details of the screening. To model this behaviour it contains a textual description mixed with the following elements in the specified order:

- one *necessary* `premiere-date` element that contains the date of the very first screening of the movie in this cinema;
- one *necessary* `premiere-room` element that contains the room number in which the premiere of the movie is shown;
- and followingly an *arbitrary number* of `screening-date` elements paired with 1 to 4 `room` elements (in this order), containing the date and the corresponding room number of the screening.

An example for valid content of the `info` element is stated followingly:

"This movie premieres on the `<premiere-date>2022-04-12</premiere-date>` in room `<premiere-room>4</premiere-room>`. The following additional screenings are offered:

On the `<screening-date>2022-04-13</screening-date>` this movie will play in rooms `<room>4</room>`, `<room>5</room>` and `<room>6</room>`.

Additionally, on the `<screening-date>2022-04-14</screening-date>` the movie will play in room `<room>6</room>`."

Element `movies`. The `movies` element has no attributes. This subtree stores all the information of various movies. To do so, the `movies` element contains an arbitrary number of `movie` elements.

Element `movie` (Child of `movies`). A `movie` element contains the following required elements in any order: a `genre` element, a `roles` element and a `name` element that contains the movie's name as a string. Additionally, the `movie` element has an attribute `mid` that adheres to the following format:

It starts with one upper case letter, followed by at a minimum one and a maximum four lower case letters and optionally a digit (0 to 9). Next, the `mid` continues with a hashtag ("#"), followed by two digits (0 to 9), which are followed by another hashtag ("#"). Finally, the `mid` ends with one additional digit (0 to 9), followed by an upper case letter and another digit (0 to 9). For example: "Spid3#22#8B5".

Element `genre`. The content of a `genre` element can be exclusively one of the strings "Drama", "Comedy" or "Action".

Element `roles`. A `roles` element contains an arbitrary amount of `support` elements and one to three `main` elements in any order (e.g. `main`, `support`, `main`, `main`, `support`). Both the `main` and `support` element contain solely a string that is equal to the name of the supporting and main role respectively. Additionally, the `main` and `support` elements contain a string attribute `actor_ref`.

Aufgabe 2 (Define keys and references `screening-plan.xsd`) *[2 Punkte]*

Add the following keys to your schema:

- A global key `movieNameKey` over the `name` element of `movie` (child of `movies`) elements.
- A global key `movieIdKey` for the attribute `mid` of `movie` (child of `movies`) elements.
- A global key `actorKey` over `aid` of `actor` elements .

Now add the following key references to your schema:

- The `movie_ref` attribute of `movie` (child of `cinema`) elements references the `movieIdKey`.
- The `actor_ref` attribute of `main` and `support` (children of `roles`) elements reference the `actorKey`.

Finally, add the following uniqueness constraint to your schema:

- Remember that each `movie` element contains a `roles` element, which contains arbitrary many `support` elements and one to three `main` elements. Make sure that *locally* in each such `roles` node, there are no duplicate entries in its child `support` and `main` elements. For instance, the following example would violate the constraint:

```
<roles>
 <support actor_ref="Act1"> Ernie </support>
 <main actor_ref="Act2"> Ernie </main>
 <main actor_ref="Act3"> Bert </main>
</roles>
```

  Additionally, also the second example below violates the described constraint:

```
<roles>
 <main actor_ref="Act4"> Miss Piggy </main>
 <second actor_ref="Act1"> Ernie </second>
 <second actor_ref="Act2"> Ernie </second>
</roles>
```

  But it is allowed that "Ernie" occurs as the child of different `roles` elements.

Aufgabe 3 (Creation of the XML document `screening-plan-xsd.xml`) *[1 Punkte]*

Create the XML document `screening-plan-xsd.xml` for the schema `screening-plan.xsd`. The XML document shall satisfy the following conditions:

- Create at least 3 `actor` elements, where one actor has two middle names, one actor has one middle name, and one actor has no middle names.
- Additionally, model that at least one `actor` element has played a main and a supporting role in 2 movies (using the `main` and `support` (child of `roles`) elements).
- Create at least 3 `cinema` elements (child of `screenings`).
- Each `cinema` element (child of `screenings`) shall have at least 2 `movie` elements. Furthermore, for each of these `movie` elements, at least one `movie` shall be followed by a valid `info` element.
- Furthermore, at least one `info` element shall contain 3 `screening-date` elements in its textual content.

- Create at least 4 `movie` elements (child of `movies`), employing all possible values of the `genre` element (i.e. `genre` shall contain at least once "Drama", "Comedy", and "Action")
- Create in total at least 6 `main` and `support` elements (child of `roles`).
- Create one element per key reference of task 2 (more specifically, create elements such that each of the `movie/movies` elements has at least one `movie/cinema` referencing it, and similarly `support` and `main` elements such that each `actor` element is referenced at least once).

Make sure that your XML-Schema `screening-plan.xsd` validates your `screening-plan-xsd.xml` document. You can check this via the following command (after installing `xmllint`):

```
xmllint --schema screening-plan.xsd screening-plan-xsd.xml
```

Instructions for downloading and installing `xmllint` can be found in TUWEL.

Important: If your XML document is not well formed and valid with regards to the schema, you will receive 0 points for this task!

Document Type Definition

Now create a DTD for the above specification.

Aufgabe 4 (Creating a DTD `screening-plan.dtd`) *[2 Punkte]*

Create a Document Type Definition (DTD) `screening-plan.dtd`, which realizes the specification form Exercise 1 and 2 above. It is possible that parts of the specification are very complicated or impossible to implement in DTD. In that case make reasonable assumptions and adaptions to implement them as close as possible in the DTD.

In your exercise interview you have to be able to explain which parts are not fully realizable in a DTD (and why). **Important:** It is particularly important that you try to implement the keys and key references. On the other hand it is not necessary to explicitly implement large number ranges through explicit enumeration of all numbers (e.g., you do not have to create an enumeration of the numbers 1 through 72 to implement a "number between 1 and 72"). Enumerations that have a small range of up to 6 values should however be implemented fully in the DTD.

Important: If you submit a DTD with syntax errors, meaning it cannot be used for validation, you will receive 0 points for the task.

Aufgabe 5 (Creating the XML document `screening-plan-dtd.xml`) *[1 Punkte]*

If your previous XML document `screening-plan-xsd.xml` does not validate with your DTD, you will have to create an additional XML document `screening-plan-dtd.xml`, which contains the same data but validates for your DTD.

Make sure that your DTD `screening-plan.dtd` validates your XML document `screening-plan-dtd.xml`. You can check this using the following command (after installing `xmllint`):

```
xmllint --dtdvalid screening-plan.dtd screening-plan-dtd.xml
```

Important: If your XML document is not well-formed or valid with regards to your DTD, you will receive 0 points for this task.