Tutorial:
Define Information Model

Author: David Haasler

Table of contents

Requirements

For you to be able to work through this tutorial, you should have a basic knowledge of XML schema. If you would like to work through a tutorial on these topics first, we recommend the tutorial of w3schools.com.

- XML Schema Tutorial

Description

The **aim** of this tutorial is to learn how to develop onion.net applications with the aid of a small scenario. A quotation management will be created by way of example, enabling quotations to be created and organised.

In the first part of the tutorial, we will open a newly installed onion.net system. We will create a new module in which we will define the XML schemas for quotations and quotation collections. The onion.net Editor will then automatically generate the maintenance masks from this information.

Note: XML schema is *the* description language for XML documents. With the automatically generated maintenance masks of the onion.net Editor, the system helps you to learn XML schema one bit at a time. Thanks to the firmly embedded schema and data consistency, not even beginners can be thrown by an onion.net live system.

Signs and symbols

Boxes marked with an arrow symbol and a green border contains instruction of what to do next.

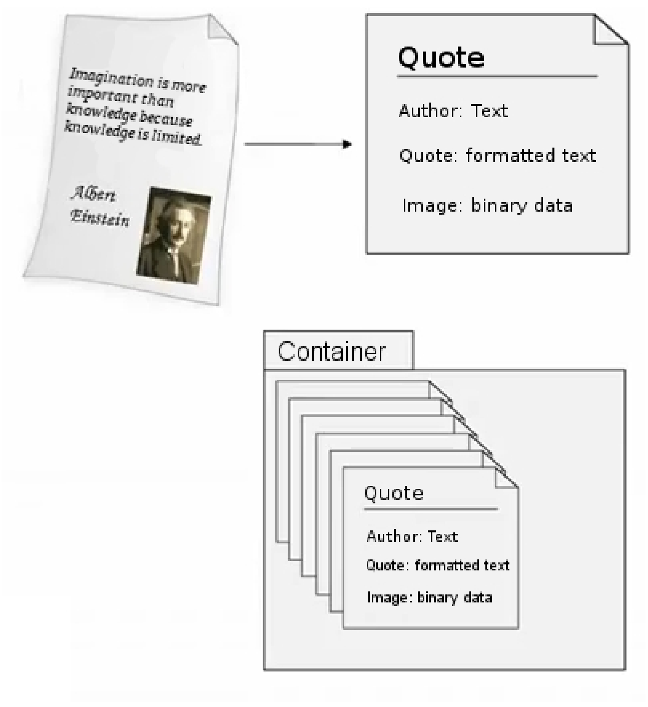
This kind of boxes contains tips and tricks.

```
Source code is shown in blue boxes.
```

1 Describing the model

We will first consider which attributes a quotation is to have. So that things do not get too complex at the beginning, a quotation shall consist of three fields in this tutorial: the author, a formatable quotation text and, optionally, a picture of the author.

It is to be possible for the quotations to be stored in a quotation container. In order to avoid cluttering when there is a large amount of quotations, it must be possible for quotations to be summarized into quotation collections. These quotation collections are in turn saved in their own container.



Description: Setup Quotation

2 Implementing the model

Once the model exists in written form, it is to be transferred into onion.net. If you have not yet installed an onion.net system, you will find helpful information in our Installation Manual. Now start your newly installed onion.net system.

If you would like to know more about the elements of the onion.net user interface, the section Building the editor will help you.

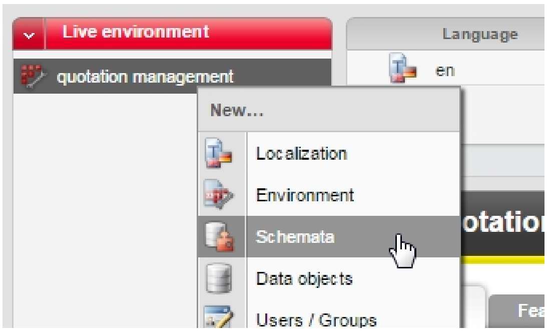
To do this, click on the "plus icon" on the top left and select *module system*. You will see 4 pre-installed modules, which are part of the core of onion.net and can neither be deactivated nor removed.

We will create a new module for the quotation management. In this module, we will carry out all work enabling quotations to be maintained. A module can be downloaded from onion.net at any time and installed in another onion.net system.

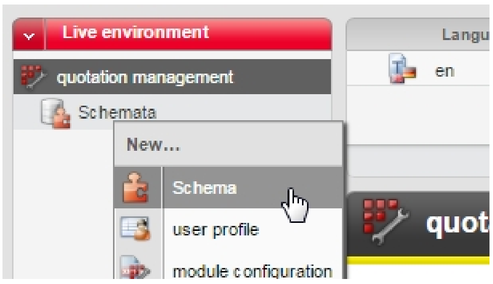
Click on the button *Create module* on the top right. Allocate a name for the module in the dialogue window (e.g. **quotation management**). If you are a manufacturer, you could select your company name for example. Clicking on *Create* will generate the module and automatically open a new tab in which the module context is loaded.

2.1 Creating schemas

In order to edit the information model we must first make it possible to do so. To do this, we need the object schemas within our module.



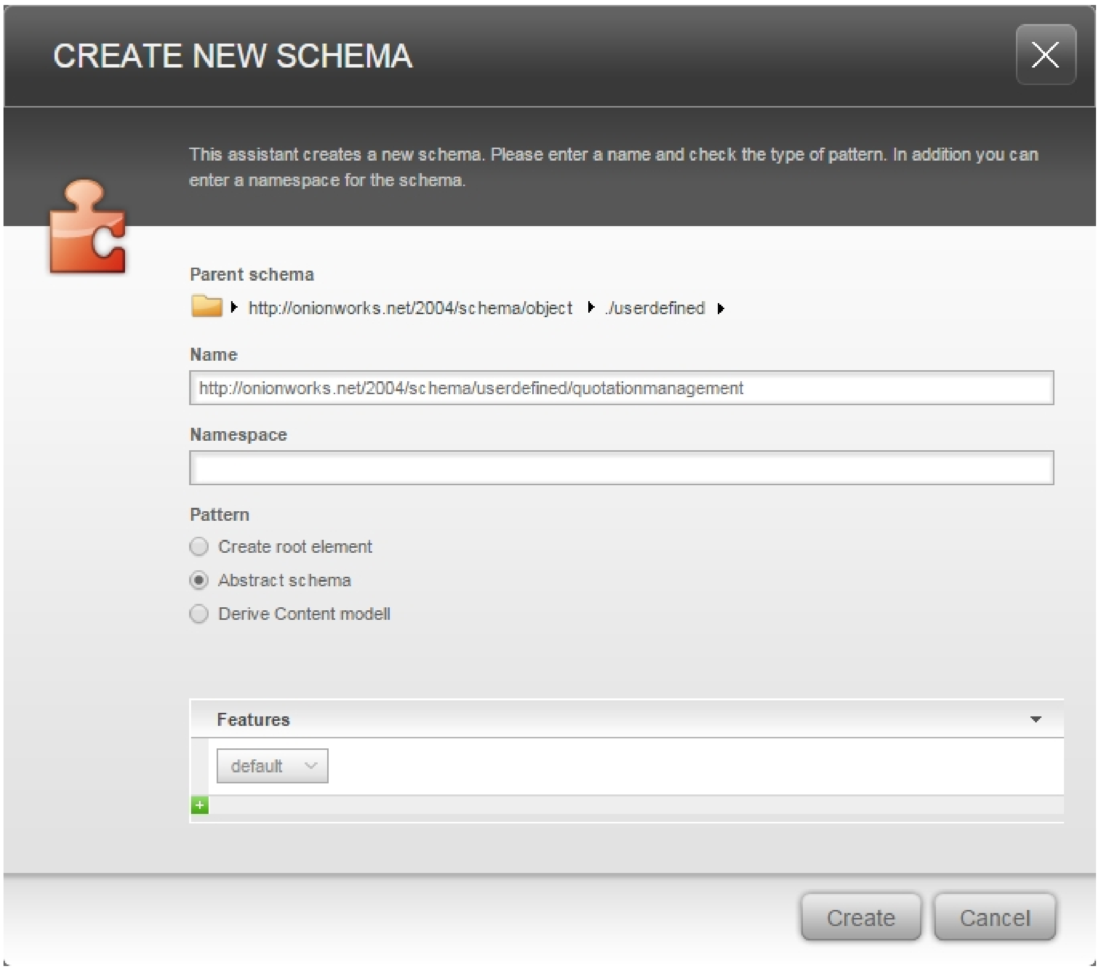
Right-click on **quotation management** (or on the module name chosen by you) and select *schemas*. In order to now create a new schema, right-click again on the element *schemas* and select the *schema* item in the context menu.



We will first define an abstract schema **http://onionworks.net/2004/schema/userdefined/quotationmanagement**. This serves as a basic type for all our quotation data types. Since it is abstract, no concrete object can be generated from it and it serves merely for grouping within the schema administration.

The onion.net type system has a hierarchical structure. User-defined schemas, such as the ones we need for the quotation management, are created below the schema *userdefined*.

Add the schema name **quotationmanagement** to the content in the name field. Under *Template*, select *Abstract schema*. Clicking on *Create* will generate the abstract schema and automatically open its editor.


CREATE NEW SCHEMA

This assistant creates a new schema. Please enter a name and check the type of pattern. In addition you can enter a namespace for the schema.

Parent schema

http://onionworks.net/2004/schema/object ▸ ./userdefined ▸

Name

http://onionworks.net/2004/schema/userdefined/quotationmanagement

Namespace

Pattern

○ Create root element
● Abstract schema
○ Derive Content modell

Features

default

Create Cancel

Below our basic type we now create a further abstract schema **quotations**. Follow the same procedure here as with the previous abstract schema. Because we are now creating a schema below the previously defined schema **quotationmanagement**, the schema defined before is already in the *name* field. Add the desired schema name **quotations** to the *name* field.

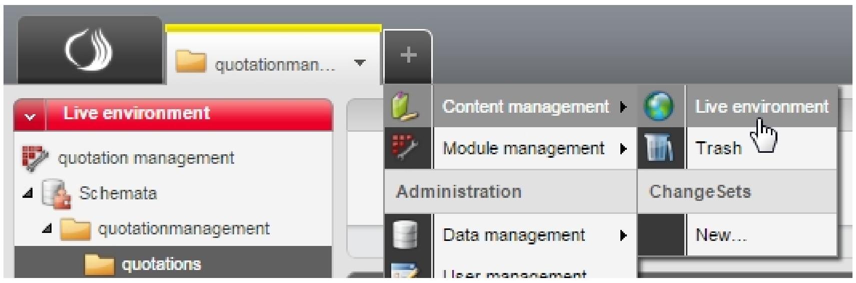
The quotation container and the quotation are to be located in this abstract schema. The latter two are concrete schemas, by which objects can also be created.

For the **quotation container**, create a new schema with the name **container** below **quotations**. Under **quotations**, create the schema **quotation** also, which will reflect our quotation object later on. Both schemas are root elements.

The template *root elements* generates a minimal XML schema with a root element. After creation, you can see the minimal schema directly in the right-hand editor window.

We now open the *Live Environment* for the first time, in which the instances of the schemas are created.

To do this, click on the *plus icon*, select *Content Management* and then *Live environment*. A new tab will open.

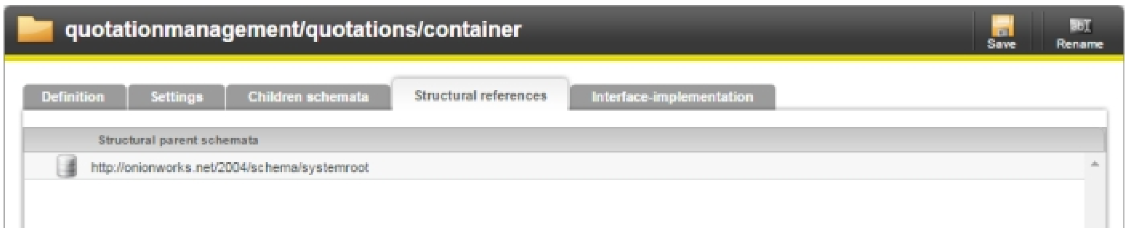


2.2 Child elements

It is to be possible for the quotation container to be created underneath the system root, characterized by the cylinder icon and the "/". Each schema defines what may be created by means of a simple child schema.

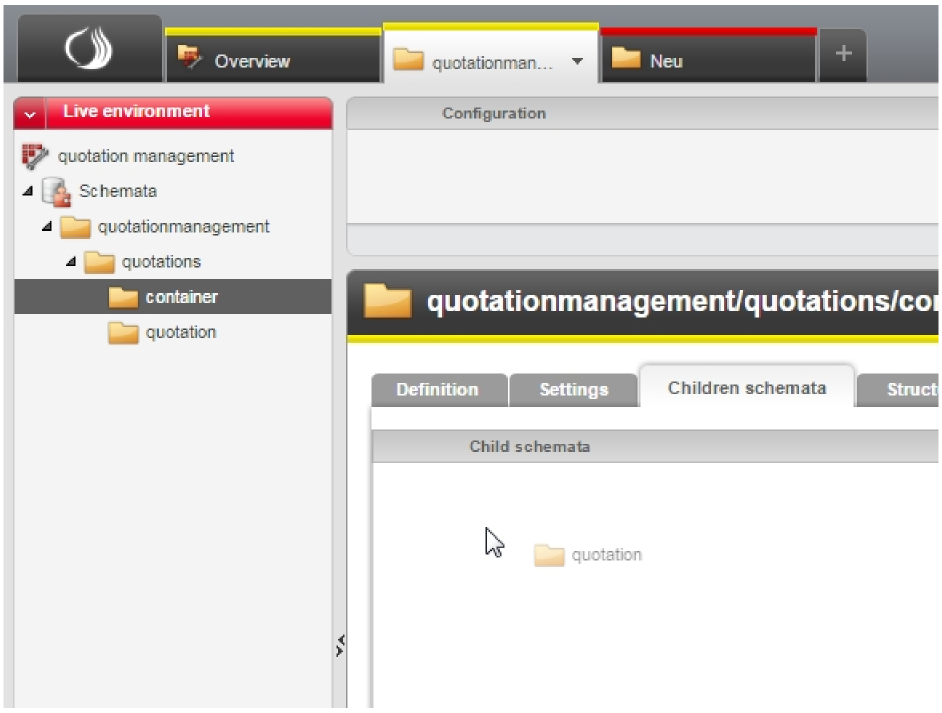
In order to display the list of creatable types, right-click on the root element. You will now see the list of possible child schemas. (In our case the list is empty. It otherwise appears above the item *Functions*).

The quotation container (**container**) is to now be inserted under the system root as a possible child schema.


quotationmanagement/quotations/container
Save
Rename
Definition
Settings
Children schemata
Structural references
Interface-implementation
Structural parent schemata
http://onionworks.net/2004/schema/systemroot

Switch to the other tab in order to return to the module context. Select the schema **container** and then open the tab **Structural references** in the object detail window. Right-click on the white section and select *Add schema* from the context menu. A so-called path chooser will appear at the bottom, via which you first choose *./system* then *. /systemroot* before then adding the path via *Select element*. Now save the change by clicking once on the *Save* symbol at the top of the object detail window.

We would like to also be able to create quotation objects in our **quotation container** later on. In order to do this we still need to define child schemas. We can do this very simply by dragging and dropping into the schemas of our module.



Select the schema **container** and then the tab *child schemas*. Now drag the schema **quotation** onto the white section in the object detail window. Save your action.

2.3 Working with object instances

You can now check this change directly by switching back into the content management and right-clicking on the system root. The system will now give us the option of creating an instance of our container.

You can create a new instance by clicking. Allocate the name **quotations**.

The instance is only actually created when you click on *Save* or *Check In*. It will then appear on the left-hand side of the tree structure.

A side note: *Check out*, *Save*, *Check In* and *Undo check out*
If you check out an object, this means that you are editing this object and no other user of the system can check it out for editing while you are doing so. Moreover, a new version of the object is also generated. You can save changes by pressing the *Save* button. If you would like to discard the changes you have made (and saved) since the last check-out, use the button *Undo ckeck out*. This will restore the state of the object as it was before being checked out. Only when you click on *Check in* will the object become editable again for other users. Clicking on *Check in* will automatically save the changes also.

Choose **quotations** as the title and return the object.

In the tree view you will now see the new object **quotations**.

Tip: You can re-sort objects in the tree view by dragging & and dropping them into their new position.

2.4 Extending the schema to include further fields

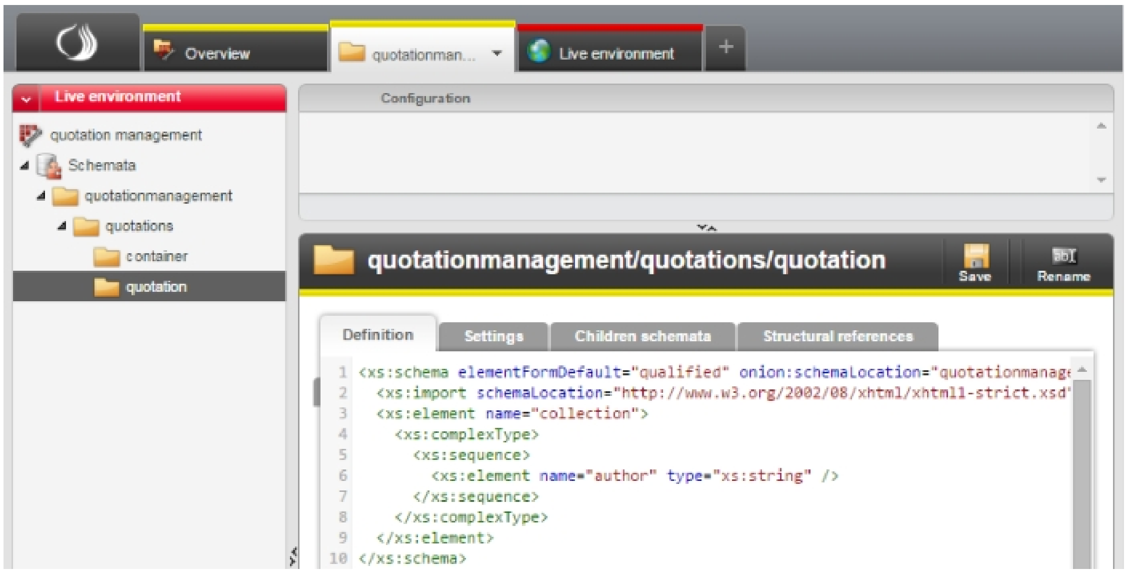
The system will now equally give us the option of creating quotations underneath the container. But do not create any just yet, since the quotation does not yet have the possibility of recording contents. We take care of this in the schema of **quotation**. We will start with a content element for recording the quotation author.

Tip: If you have not yet looked into XML schemas, now is a good time to read through a relevant tutorial. You will find a good tutorial on the pages of w3schools.org: Introduction to XML Schemas (http://www.w3schools.com/schema/schema_intro.asp)

2.4.1 String

Within the <xs:complexType> element, the empty <xs:sequence> element is deleted and replaced with the following sequence:

```
<xs:sequence>
  <xs:element name="author" type="xs:string" />
</xs:sequence>
```



```
<xs:schema elementFormDefault="qualified" onion:schemaLocation="quotationmanage
  <xs:import schemaLocation="http://www.w3.org/2002/08/xhtml/xhtml1-strict.xsd'
  <xs:element name="collection">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="author" type="xs:string" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Now save the schema. In the data section (in the content management), a suitable maintenance option will automatically appear for each schema element. In this case you will see the new field **.author**, which you can enter a string into. Do not save this test yet, as we have yet to add the two further fields.

2.4.2 Rich text field (flow)

The next field that requires the quotation is the quotation text itself. The quotation text itself is to be able to contain formatting. For this purpose, we use the flow data type out of XHTML Strict.

Extend the <xs:sequence> to include a further element:

```
<xs:element name="quote" type="xhtml:Flow" />
```

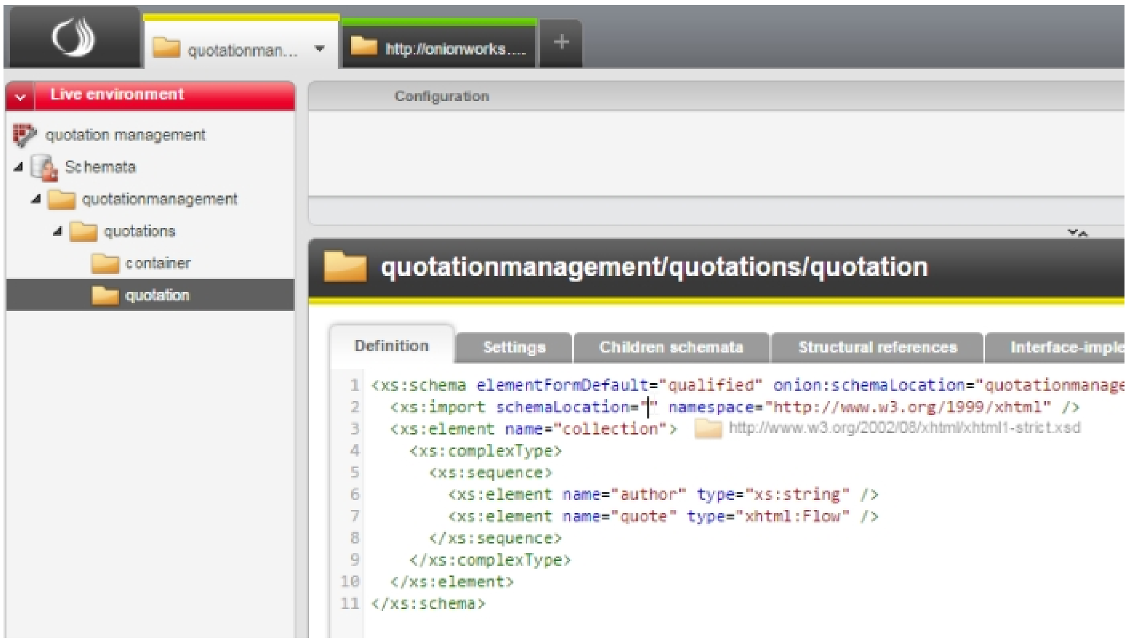
Since the namespace *xhtml* has not yet been defined, we will now provide the necessary information to the schema. To do this, we extend the <xs:schema> element to include the namespace xhtml:

Tip: You will also find further information about namespaces at w3schools.org: XML Namespaces (http://www.w3schools.com/xml/xml_namespaces.asp)

The XHTML schema is already in the system. Data types from other schemas can be easily imported. You will find the XHTML schema in the schema of the information server core module.

Open the context menu of the root in the productive environment and select *Advanced* and *Navigate to schema*.

There also further schemas there, of which we will need the XLINK schema in a moment.



```
<xs:schema elementFormDefault="qualified" onion:schemaLocation="quotationmanage
  <xs:import schemaLocation="" namespace="http://www.w3.org/1999/xhtml" />
  <xs:element name="collection">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="author" type="xs:string" />
        <xs:element name="quote" type="xhtml:Flow" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Import the XHTML schema by means of the instruction <xs:import schemaLocation=““ namespace=““ /> as the first element within xs:schema. Now drag the schema *http://www.w3.org/2002/08/xhtml1-strict.xsd* from the tab of the core module onto the tab of the own module. You can then release the schema path in the attribute *schemaLocation*. Enter *http://www.w3.org/1999/xhtml* into the attribute *namespace*.

Overall, the schema will now look as follows after this step:

```
<xs:schema elementFormDefault="qualified"
onion:schemaLocation="quotationmanagement/quotations/quotation">
  <xs:import schemaLocation="http://www.w3.org/2002/08/xhtml/xhtml1-strict.xsd"
namespace="http://www.w3.org/1999/xhtml" />
  <xs:element name="quotation">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="author" type="xs:string" />
        <xs:element name="quote" type="xhtml:Flow" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

In the above source code, the indication of the namespace for onion is still missing in the schema tag. Add *xmlns:onion="http://onionworks.net/2004/schema*. Now save the changes.

If you now create a new quotation, you will see a rich text editor open underneath the **author** field, which the quotation text (quote) can be entered into together with formatting. Do not save the quotation yet either, since we are going to insert the third field straight away.

2.4.3 Picture (binaryReference)

In the next step we will add an optional picture to the quotation. The necessary data type (binaryReference) is defined in the XLINK schema.

To do this, extend the <xs:sequence> to include the following element:

```
<xs:element
  name="image"
  type="xlink:binaryReference"
  minOccurs="0"
/>
```

The attribute *minOccurs* means here that an object created by this type must have at least as many elements of this type as indicated here. The value *0* makes this element optional.

Because of the new namespace we need a further import instruction. Just like the XHTML schema, the XLINK schema is located in the schema folder of the core module and can be dragged & dropped into the attribute *schemaLocation* and *namespace*:

```
<xs:import schemaLocation="http://www.w3.org/1999/xlink"
namespace="http://www.w3.org/1999/xlink" />
```

Our quotation data type is complete. The complete schema now looks as follows:

```
<xs:schema elementFormDefault="qualified"
onion:schemaLocation="http://example.com/2009/showcase/quotations/quotation">
  <xs:import schemaLocation="http://www.w3.org/2002/08/xhtml/xhtml1-strict.xsd"
namespace="http://www.w3.org/1999/xhtml" />
  <xs:import schemaLocation="http://www.w3.org/1999/xlink"
namespace="http://www.w3.org/1999/xlink" />
  <xs:element name="quotation">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="author" type="xs:string" />
        <xs:element name="quote" type="xhtml:Flow" />
        <xs:element
                name="image"
                type="xlink:binaryReference"
                minOccurs="0"
        />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

2.5 Creating quotations

We now enter three quotations. In this tutorial we will use these three quotations:

Name: Quote Albert Einstein
Author: Albert Einstein
Quotation: Imagination is more important than knowledge. For knowledge is limited.

Name: Quote Konrad Adenauer
Author: Konrad Adenauer
Quotation: All of us live under the same sky, but we don't all have the same horizon.

Name: Quote Thomas Carlyle
Author: Thomas Carlyle
Quotation: The greatest of all faults, I should say, is to be conscious of none.

Many web pages offering free downloadable images of this authors to add to our quotations. They can be found on the web using relevant search engines.

To do this, switch to the content management and right-click on your **quotations** element. Under the item *New...* you can create a new object of the schema **quotation**. Fill out all fields and upload a suitable picture. When you are finished with a quotation, click *Check in*.

2.6 Further schemas

We now need the data types in order to create quotation collections. For this purpose, switch back to the schemas of the **quotation management** module. The procedure is now fairly similar. We need a container (**container**) for the collections (**collection**). We group the data types for collections below an abstract type **collections**.

Tip: Since we are now working with quite a number of schemas, it makes sense to review the data model:

Data type	Kind (option)	Schema location (place)
Basic type	Abstract schema	quotationmanagement
Quotations	Abstract schema	quotationmanagement/quotations
container	Root element	quotationmanagement/quotations/container
Quotation	Root element	quotationmanagement/quotations/quotation
Collections	Abstract schema	quotationmanagement/collections
container	Root element	quotationmanagement/collections/container
Collection	Root element	quotationmanagement/collections/collection

For this purpose, create a new abstract schema **collections** below **quotationmanagement**. And in this schema, create the root element **container**.

It is to be possible to create the container again below the system root. We now add a parent schema.

To do this, click on the tab *structural references* and add the schema *http://onionworks.net/2004/schema/systemroot* there. Save the change. Create a new object **quotation collections** under the *system root* in the *content management* and return the object.

We will now create the data type for the collection. We then add this to the child schemas of the container.

In the schemata of the module, you now right-click on the schema **collections** and create a new root element **collection**. So that a container can be created below the collections (**container**), select **container**, switch to the tab *Child schemata* and drag the new schema **collection** into the white section before saving the change.

The collection is to refer to at least two quotations. XML schema makes this possible by indicating *minOccurs* and *maxOccurs*.

For this purpose, insert the following object reference within the <xs:sequence> element that is still empty:

```
<xs:element
  name="quotation"
  type="xlink:objectReference"
  minOccurs="2"
  maxOccurs="unbounded"
/>
```

The value *unbounded* means unlimited.

Since we are again using an XLINK, we need to import its data type. We do this in the second line of the schema by inserting the tag:

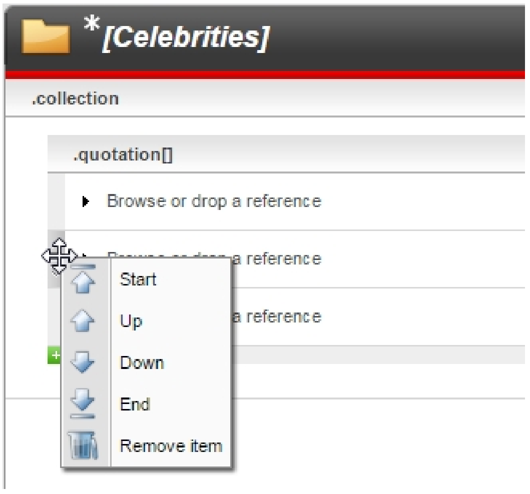
```
<xs:import schemaLocation="" namespace="" />
```

In order to save typing, you can now once again drag the XLINK schema from the schemas of the system root and into the attributes *schemaLocation* and *namespace*. Now save the schema.

Concrete collections can now be created below the container **quotation collections** that has just been created. Now create such a collection.

Then switch to the detail view and right-click on the container **quotation collections**. Create a new collection and give it the title **Celebrities**.

You will now see a section with the name **.collection**, into which two references can be dragged. Underneath this there is the function for adding a new **quotation** (small green plus icon). To the right of each reference there is a function box which will open a context menu when clicked on. In this context menu you have the option of sorting or deleting the entries.



Now drag the quotations of Albert Einstein and Konrad Adenauer into the two fields shown. Then click on the green plus icon and incorporate the Thomas Carlyle quotation into the collection as well. Now change the sorting using the function boxes on the right-hand side and place Konrad Adenauer at the beginning. Then return the object.

Lastly, we will add the **interval** property to the schema. This is to enable an editor to control how long a quotation is displayed for, and may be of interest for the later output of the quotation on a website for example.

Switch back to the schemas of the module and select the **collection** below **collections**. In the definition, add an attribute element after the closing <xs:sequence> element.

```
<xs:attribute
  name="interval"
  type="xs:positiveInteger"
  default="5"
/>
```

Now switch back to your collection **Celebrities**. You will now see, above the three quotations, the new attribute **interval** with the pre-set value **5**.

You've done it!

In the next part we will look at utilising our quotation collection.