CoCeSo
Prototyp CakePHP
Schnittstellendokumentation

Daniel Rohr

20. August 2013

Inhaltsverzeichnis

1 Allgemeine Beschreibung der CRUD-Schnittstellen

Die Datenausgabe kann in *HTML* (default) oder *JSON* erfolgen. Zur Erzeugung von JSON wird der Header `Accept: application/json` gesetzt oder an das Ende der URL `.json` angefügt. Für JSON wird ein Objekt als Root-Element ausgegeben, das als Variablen die einzelnen Datenobjekte enthält. In der Folge wird die Bedienung der Schnittstellen über JSON beschrieben.

1.1 Create

Die Erzeugung eines Eintrages erfolgt durch die Methode *add* im jeweiligen Controller, die durch `/controller/add` via *POST* aufzurufen ist. Die Daten werden als POST-Body im Array `data[Model]` übergeben. Dabei darf kein Feld für die ID vorhanden sein, ansonsten wird der *HTTP-Fehler 400 Bad Request* zurückgegeben.

Die Antwort für JSON ist einerseits ein boolscher Wert `success`. Im Erfolgsfall wird weiters die ID des eingefügten Eintrags im Feld `id` ausgegeben. Falls Fehler aufgetreten sind wird ein Objekt `errors` ausgegeben, das für jedes fehlerhafte Datenfeld ein Array der fehlgeschlagenen Validierungen enthält.

Request

```
POST /controller/add HTTP/1.1
Accept: application/json

data[Model][column1]=value1
data[Model][column2]=value2
...
```

Antwort bei Erfolg

```
{"success":true,"id":"1"}
```

Antwort bei Fehler

```
{"success":false,"errors":{"column1":["error1","error2"],...}}
```

1.2 Read

1.2.1 Auslesen aller Einträge

Über die *index*-Methode des Controllers, die neben `/controller/index` auch direkt durch `/controller` erreichbar ist, werden die vorhandenen Einträge aufgelistet. Die JSON-Ausgabe unterscheidet sich dabei leicht von der oben beschriebenen. Das Root-Element ist ein Array, das die einzelnen Einträge als Objekte enthält.

Diese Schnittstelle muss noch um Filtermöglichkeiten erweitert werden und sollte in der Ausgabe Informationen zur Seitenschaltung enthalten.

1.2.2 Auslesen eines bestimmten Eintrags

Ein bestimmter Eintrag kann durch `/controller/view/id` ausgelesen werden. Dabei werden auch die dazu in Beziehung stehenden Elemente ausgegeben. Die Anforderung eines nicht existierenden Eintrages erzeugt den *HTTP-Fehler 404 Not Found*.

Request

```
GET /controller/view/1 HTTP/1.1
Accept: application/json
```

Antwort

```
{"Model":{"id":"1","field1":"value1","field2":"value2",...},
 "AssociatedModel1":[{"id":"1","fieldx":"valuex",...},{...}]}
```

1.3 Update

Die Änderung eines existierenden Eintrages erfolgt durch Aufruf der *edit*-Methode über POST. Die Daten werden dabei wie beim Hinzufügen eines Eintrages übergeben. Die ID kann wie beim Auslesen durch den Aufruf mittels */controller/edit/id* oder im POST-Body mit den restlichen Daten übergeben werden. Dabei ist zu beachten, dass ein *HTTP-Fehler 400 Bad Request* erzeugt wird, wenn beide Werte gesetzt sind, aber nicht übereinstimmen. Existiert der Eintrag nicht wird wieder der *HTTP-Fehler 404 Not Found* zurückgegeben. Die Ausgabe entspricht der *add*-Methode.

Request: ID via URL

```
POST /controller/edit/1 HTTP/1.1
Accept: application/json

data[Model][column1]=newvalue1
...
```

Request: ID via POST

```
POST /controller/edit HTTP/1.1
Accept: application/json

data[Model][id]=1
data[Model][column1]=newvalue1
...
```

Antwort bei Erfolg

```
{"success":true,"id":"1"}
```

Antwort bei Fehler

```
{"success":false,"errors":{"column1":["error1","error2"],...}}
```

1.4 Delete

Im Prototyp ist für keine der Entitäten die Möglichkeit der Löschung eines Eintrags vorgesehen.

2 Liste aller Schnittstellen

	URL	Methode
3.2	`/incidents/add`	`POST`
3.3	`/incidents/index`	`GET`
3.3	`/incidents/view`	`GET`
3.4	`/incidents/edit`	`POST`
4.2	`/units/add`	`POST`
4.3	`/units/index`	`GET`
4.3	`/units/view`	`GET`
4.4	`/units/edit`	`POST`
5.2	`/incidents_units/add`	`POST`
5.4	`/incidents_units/edit`	`POST`

In der Folge kennzeichnet *r* lesbare Felder, *rw* les- und schreibbare Felder und * Pflichtfelder. Fettgedruckte Werte werden als Standardwert gewählt, wenn keiner angegeben ist.

3 Entität *Incident*

3.1 Felder

`id`	`<unsigned int>`	`r`	Primary Key, beim Einfügen automatisch generiert
`created`	`<datetime>`	`r`	Erstellungszeit, automatisch gesetzt
`modified`	`<datetime>`	`r`	Letzte Änderung, automatisch gesetzt
`finished`	`<datetime>`	`r`	Abschluss des Vorfalls automatisch mit `status=4` gesetzt
`type`	`<tinyint>`	`rw`	Typ des Vorfalls: 1-Info, 2-Verlegung, 3-Auftrag/Einsatz
`priority`	`<boolean>`	`rw`	Einsatzmäßige Durchführung/Verwendung der Sondersignale
`text`	`<text>`	`rw`	Einsatztext: Freitext
`comment`	`<text>`	`rw`	Zusätzlicher Kommentar: Freitext
`status`	`<tinyint>`	`rw`	Status: 0-Neu, 1-Offen, 2-Disponiert, 3-in Arbeit, 4-Abgeschlossen

3.2 Create

Die Erstellung erfolgt durch POST an `/incidents/add`. Zulässige Werte sind:

`type`	*	1,2,3
`priority`		0,1
`text`		beliebig
`comment`		beliebig
`status`		**0**,1,2,3,4

Für die Status *Disponiert* und *in Arbeit* gilt dabei die Einschränkung, dass mindestens eine Einheit (siehe 5.1) *Zugewiesen/ZBO* resp. *BO/ZAO/AO* sein muss. Setzen des Status *Abgeschlossen* bewirkt, dass sämtliche zugewiesenen Einheiten auf *Nicht zugewiesen* gesetzt werden.

Request

```
POST /incidents/add HTTP/1.1
Accept: application/json

data[Incident][type]=2
data[Incident][text]=Einsatztext
```

3.3 Read

Einzelne Einträge enhalten neben dem `Incident`-Element noch ein Array `Unit`, das sämtliche zugeordneten Einheiten enthält. Zu jeder zugeordneten Einheit wird auch der Status der Zuordnung ausgegeben.

Request

```
GET /incidents/view/1 HTTP/1.1
Accept: application/json
```

Antwort

```
{
  "Incident":{
    "id":"1",
    "created":"2013-07-26 10:00:00",
    "modified":"2013-07-26 12:00:00",
    "finished":null,
    "type":"2", "priority":false,
    "text":"Einsatztext", "comment":"", "status":"3"
  },
  "Unit":[{
    "id":"2", "name":"Trupp 2", "short":"TRP2",
    "type":"Trupp", "status":"2",
    "IncidentsUnit":{
      "id":"1", "incident_id":"1", "unit_id":"2",
      "status":"3", "modified":"2013-07-26 18:00:00"
    }
  }]
}
```

3.4 Update

Die Bearbeitung erfolgt durch POST an `/incidents/edit`. Für die Werte gelten dabei die selben Regeln wie in 3.2, wobei `type` nicht verpflichtend angegeben werden muss. Der Status *Neu* kann nicht mehr gesetzt werden, wenn zuvor bereits ein anderer gesetzt war.

4 Entität *Unit*

4.1 Felder

`id`	`<unsigned int>`	`r`	Primary Key, beim Einfügen automatisch generiert
`name`	`<varchar(30)>`	`rw`	Funkrufname der Einheit
`short`	`<varchar(30)>`	`rw`	Abkürzung der Einheit
`type`	`<enum>`	`rw`	Typ der Einheit: 'Trupp', 'KFZ'
`status`	`<tinyint>`	`rw`	Status: 0-AD, 1-NEB, 2-EB, 3-Bereitschaft

4.2 Create

Die Erstellung erfolgt durch POST an `/units/add`. Zulässige Werte sind:

`name`	*	eindeutig
`short`	*	eindeutig
`type`	*	'Trupp','KFZ'
`status`		**0**,1,2,3

4.3 Read

Einzelne Einträge enhalten neben dem `Unit`-Element analog zu 3.3 noch ein Array `Incident`, das sämtliche zugeordneten Einheiten enthält.

Request

```
GET /units/view/2 HTTP/1.1
Accept: application/json
```

Antwort

```
{
  "Unit":{
    "id":"2", "name":"Trupp 2", "short":"TRP2",
    "type":"Trupp", "status":"2",
  },
  "Incident":[{
    "id":"1",
    "created":"2013-07-26 10:00:00",
    "modified":"2013-07-26 12:00:00",
    "finished":null,
    "type":"2", "priority":false,
    "text":"Einsatztext", "comment":"", "status":"3"
    "IncidentsUnit":{
      "id":"1", "incident_id":"1", "unit_id":"2",
      "status":"3", "modified":"2013-07-26 18:00:00"
    }
  }]
}
```

4.4 Update

Die Bearbeitung erfolgt durch POST an `/units/edit`. Für die Werte gelten dabei die selben Regeln wie in 4.2, wobei es keine Pflichtfelder gibt.

5 Entität *IncidentsUnit*

Diese Entität verknüpft Vorfälle und Einsätze in einer n:n-Beziehung.

5.1 Felder

`id`	`<unsigned int>`	`r`	Primary Key, beim Einfügen automatisch generiert
`incident_id`	`<unsigned int>`	`rw`	Foreign key zum Vorfall
`unit_id`	`<unsigned int>`	`rw`	Foreign key zur Einheit
`status`	`<tinyint>`	`rw`	Status: 0-Nicht zug., 1-Zug., 2-ZBO, 3-BO, 4-ZAO, 5-AO
`modified`	`<datetime>`	`r`	Letzte Änderung, automatisch gesetzt

5.2 Create

Die Erstellung erfolgt durch POST an `/incidents_units/add`. Die Felder `incident_id` und `unit_id` müssen auf existierende Vorfälle bzw. Einheiten verweisen. Jedes Paar `Incident-Unit` darf nur einmal vorkommen. Zulässige Werte für den Status sind 0,**1**,2,3,4,5.

5.3 Read

Es gibt keine Read-Methoden.

5.4 Update

Die Bearbeitung erfolgt durch POST an `/incidents_units/edit`. Nur der Status kann verändert werden, zulässige Werte sind wie in 5.2.