Erstellen von Containern mit Zugriff auf ein Verzeichnis via LDAP

Dieses Tutorial beschreibt die einzenlen Schritte zum erstellen eines Containers, der auf Daten aus einem LDAP-Verzeichs liest.

Table of contents

1. Anforderungen

Dieses Dokument setzt vorraus, dass Galaxy bereits auf einem System installiert ist, eine Beschreibung der Installationsprozedur findet sich auf der Galaxy Webseite, sowie im Installationspaket, dass dort ebenfalls heruntergeladen werden kann.

Für die Installation werden die folgenden Komponenten benötigt:

- Ein Applicationserver (z.B. Jakarta Tomcat oder IBM's WebSphere)
- Eine Java Laufzeit Umgebung ab Version 1.4
- Eine Datenbank zum Speichern der Konfiguration, z.B. MySQL.

2. Erstellen des Connectors

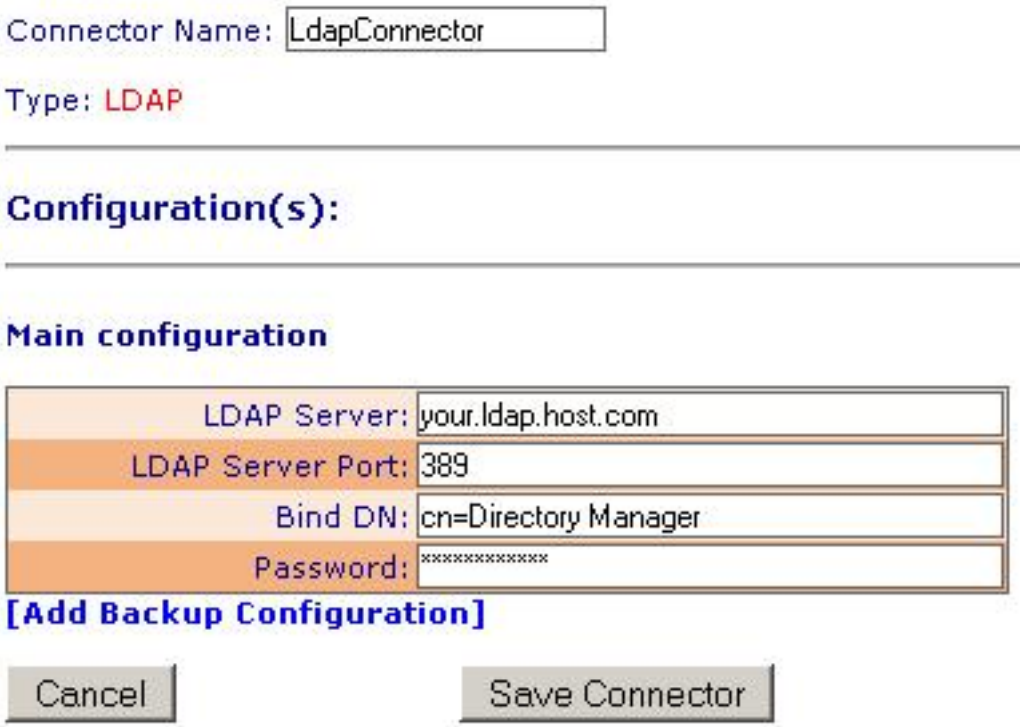
Ein Connector stellt die unterste Ebene in der Architektur von Galaxy da und repräsentiert eine Quelle, die Daten bereitstellt. Connectoren können in verschiedenen Instanzen existieren. So können z.B. zwei Instanzen des LDAP-Connectors aus zwei verschiedenen Verzeichnissen unterschiedliche Daten auslesen. Die beiden Instanzen operieren dann vollkommen unabhängig.

Um Anfragen gegen einen Verzeichnisdienst über LDAP durchzuführen, wird also ein Connector vom Typen "LDAP" benötigt, dieser muss zunächst erstellt werden.

Hierzu muss der Menüpunkt "Connectors" und anschließend "Create Connector" Ausgewählt werden. Es zeigt sich die folgende Seite:

Erstellen des Connectoren

In der gezeigten Liste muss der Punkt "LDAP" selektiert werden und dies muss anschließend mit dem Button "Create!" bestätigt werden. Dies führt zur Konfigurationsseite des Connectoren, auf der seine Eigenschaften bearbeitet werden können:



Konfiguration des Connector

Zunächst muss ein Name für diesen Connector vergeben werden, in diesem Tutorial nennen wir Ihn schlicht "LdapConnector".

Anschließend müssen die Verbindungseigenschaften spezifiziert werden. Diese sind:

- **LDAP Server** - *IP-Adresse oder Hostname des LDAP Servers*
- **LDAP Server Port** - *Port, auf dem der Server Anfragen entgegenimmt*
- **Bind DN** - *Benuzername für die Authentifizierung am Server (Optional)*
- **Password** - *Passwort für die Authentifizierung (Optional)*

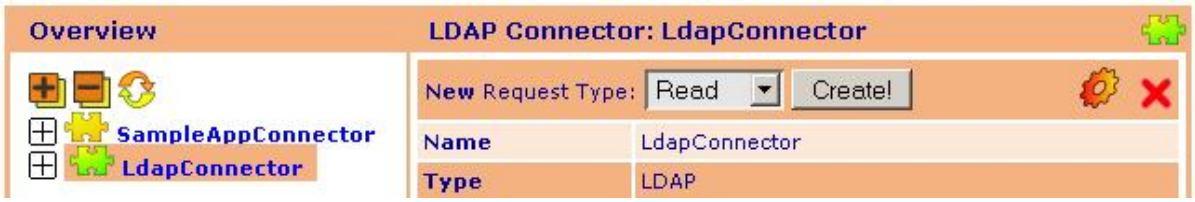
*Werden **Bind DN** und **Passwort** nicht bestückt, so wird versucht ein anonymes Binden durchzuführen.*

Mit Hilfe des Links "[Add Backup Configuration]" kann eine Failover-Kofiguration hinzugefügt werden, die aktiv wird,wenn die Hauptkonfiguration des Connectors nicht verfügbar ist. Dies macht allerdings nur Sinn, wenn dieser zweite Server die selben Daten bereitstellt

Ist die Konfiguration abgeschlossen, kann der Connector mit Hilfe des Buttons "Save

Connector" gespeichert werden.

Nach Auswahl des Menüpunktes "Connectors" ist der neu erstellte Connector nun in der Liste aller definierten Connectoren aufgeführt:



Liste der Connectoren

Über das Zahnrad Symbol kann die Konfiguration erneut Bearbeitet und gespeichert werden.

3. Konfiguration eines Requests

Ein Request ist das Bindeglied zwischen Connectoren und Containern. Er beschreibt die Parameter der Anfrage, die gegen den Connector gestellt wird, und stellt die Antwort in Form von Ausgabefeldern bereit, die dann in eigene Container verpackt werden können.

Ein Connector kann mehrere Requests bereitstellen, die verschiedene Formen von Abfragen realisieren.

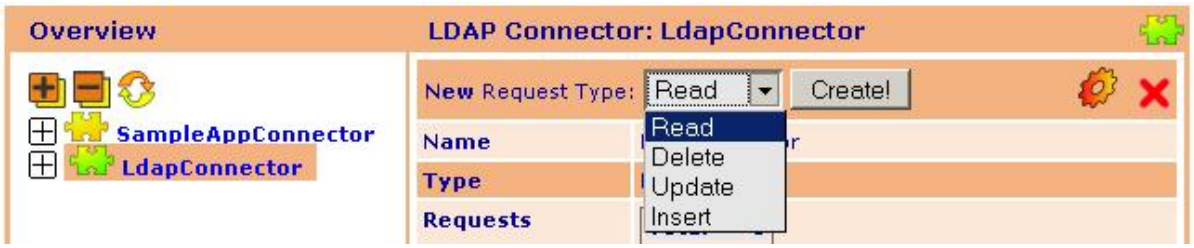
Um nun einen Request für unseren LdapConnector zu erstellen, muss der Menüpunkt "Connectors" angewählt werden. In der Baumstruktur wird nun der eben erstellte Connector "LdapConnector" ausgewählt.

Da noch keine Requests konfiguriert wurden, ist diese Liste zunächst leer

In der gezeigten Auswahlliste stehen nun verschiedene Typen von Requests zur Auswahl:

- **Read** - Erstellt einen *lesenden* Request
- **Delete** - Erstellt einen Request, der bestehende Daten *löscht*
- **Update** - Erstellt einen Request, der bestehende Daten *modifiziert*
- **Insert** - Erstellt einen Request, der neue Daten *einfügt*

In diesem Tutorial soll ein lesender Request konfiguriert werden, hierzu muss der Eintrag **Read** ausgewähtlt und mit dem Button "Create!" bestätigt werden:



Liste der Requests

Dies führt zur Konfigurationsseite des neuen Request:

Request Konfiguration 1

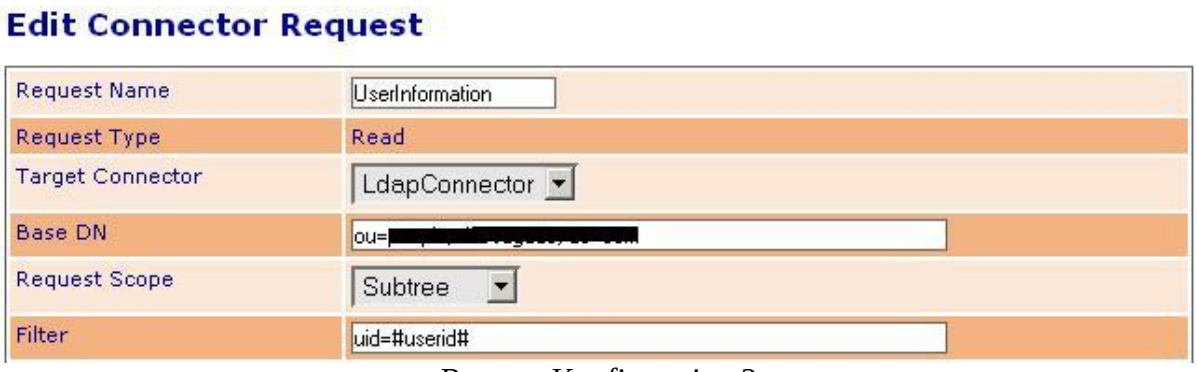
Hier eine kurze Erläuterung der einzelnen Felder:

- **Request Name** - ein Name für den Request
- **Request Type** - der vorher ausgewählte Request Type
- **Target Connector** - Der Connector auf den dieser Request abzielt
- **Base DN** - Das Basisobjekt für den Request

- **Request Scope** - Die Suchstrategie innerhalb der Hirachie:
  - **Object** - Durchsucht nur das in *Base DN* spezifizierte Objekt
  - **One Level** - Sucht eine Ebene unterhalb des *Base DN's*
  - **Subtree** - Durchsucht den gesamten Baum unterhalb von *Base DN*
- **Filter** - Der Filter, der bei der Suche angewendet werden soll
- **Error on empty result** - Ist diese Checkbox ausgewählt, so wird ein Fehler gemeldet, wenn die Anfrage keine Ergebnise zurückliefert
- **Provided Table** - Durch einen Klick auf den Link "Provide as Table" kann das Ergebnis auch in Form einer *Tabelle* zurückgeliefert werden. Dies macht Sinn, wenn ein Request mehrere Objekte zurückliefern soll.
- **Sample Data** - Hier kann eine Testanfrage durchgeführt werden, um Beispieldaten zu erhalten.
- **Output Fields** - Hier werden die Attribute Ausgabefeldern zugeordent, die der Request dann zur Verfügung stellt.

3.1. Basisdaten

Als erstes werden die Basisdaten wie folgt eingetragen:



Request Konfiguration 2

Base DN sowie *Filter* müssen dabei an die Struktur des entsprechenden Verzeichnises angepasst werden

Im Filter-Feld können Variablen genutzt werden, die zur Laufzeit der Anfrage ersetzt werden. Um eine Variable einzusetzen, muss der Name in zwei "#"-Zeichen eingefasst sein. (z.B. #userid# oder #kundennummer#, siehe Screenshot)

Die so definierten Variablen können später mit einem Eingabefeld eines Conatiners verknüpft werden und sind somit für jede Anfrage spezifizierbar.

Beispiel: In einem Verzeichnis stehen Informationen zu Kunden einer Firma. Jeder Kunde

hat eine eindeutige Kennnummer, anhand dieser kann er identifiziert werden . Möchte nun eine Applikation nur einen speziellen dieser Datensätze lesen, so muss bei der XML-Anfrage eine Kundennummer spezifiziert werden. Bei der eigentlichen Verzeichnisabfrage wird nun eine Variable (z.B. #userid#) durch den mitgelieferten Wert ersetzt, wodurch nur der gewünschte Datensatz gefunden wird. Dieser wird nun wieder in XML umgesetzt und zurück an die Applikation gesendet.

3.2. Daten als Tabelle bereitstellen

Da wir die das Ergebnis dieses Request auch als Tabelle bereitstellen wollen, sollte nun der Link "Provide as Table" betätigt werden:

Request Konfiguration 3

Es wird ein Textfeld bereitgestellt, in dem ein Name für die Tabelle eingetragen werden kann, wir nennen sie "User List":

Request Konfiguration 3

Eine Tabelle wird als eigenständiges Ausgabefeld bereitgestellt, und kann als solches in einen Container eingebunden werden. Um dieses Feld wieder zu entfernen, kann der Link "Remove Table" genutzt werden.

3.3. Durchführen eines Beispiel Requests

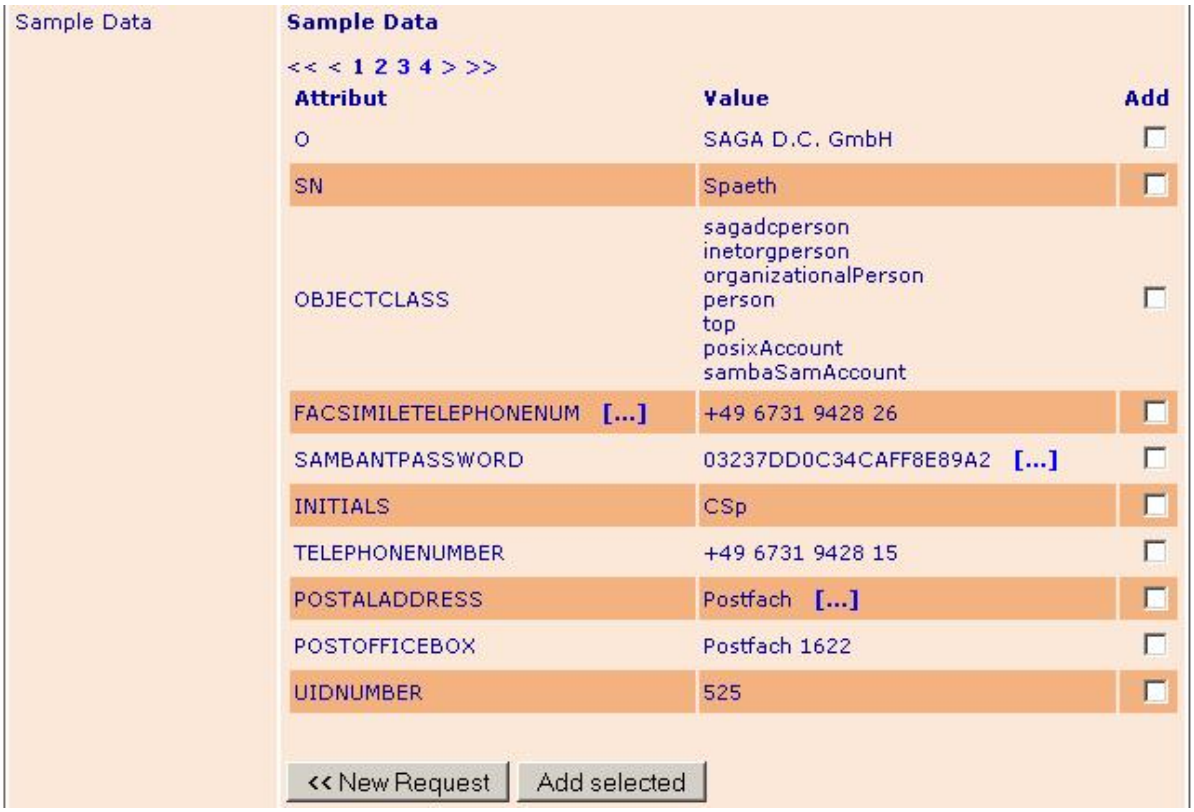
Im Abschnitt "Sample Data" wird nun für jede Variable, die innerhalb des Filters gefunden wurde (in unserem Fall "#userid#") ein Textfeld zur Verfügung gestellt, das mit einem Wert gefüllt werden kann:

Request Konfiguration 3

Wenn der Filter verändert wird, kann der Button "Update Fields" genutzt werden, um erneut nach Variablen zu suchen

Wie auf der Abbildung zu sehen ist, wurde der Wert "cspaeth" eingetragen. Über den Button "Run Request >>" wird nun eine Anfrage gegen den entsprechenden Ldap-Server ausgelöst. Der Wert aus dem Textfeld "userid" wird hierbei an der Stelle im Filter eingesetzt, an dem die Variable userid steht ("#userid#").

Das erste Objekt, das diese Anfrage zurückliefert, wird nun in Form einer Tabelle dargestellt:



Request Konfiguration 3

Diese Enthält die folgenden Spalten:

- **Attribut** - Der Name des Attributes
- **Value** - Der Wert des Attributes *Besitzt ein Attribut mehr als einen Wert, werden diese durch Zeilenumbrüche getrennt, siehe Attribut "**OBJECTCLASS**"*
- **Add** - Mit Hilfe dieser Checkbox kann ein Attribut bequem als Ausgabefeld für diesen Request bereitgestellt werden. *Dies wird im nächsten Schritt beschrieben.*

Die Tabelle ist in mehrere Seiten unterteilt die mit Hilfe der Links oberhalb der Tabelle durchgeblättert werden können.

Um einen neuen Request mit neuen Werten durchzuführen, dient der Buttion "New Request", dieser führt zurück zum Ausgangszustand.

3.4. Definieren der Ausgabefelder

In der nächsten Sektion wird definiert, welche Felder dieser Request zurückliefern soll. Die Tabelle ist zunächst leer (*Es wurden noch keine Felder definiert*):

Request Konfiguration 3

Um nun Ausgabefelder zu definieren, können die gewünschten Attribute im Abschnitt "Sample Data" mit Hilfe der Checkbox in der Spalte "Add" selektiert werden:

Request Konfiguration 3

Durch betätigen des Buttons "Add Selected" werden die ausgewählten Attribute übernommen:

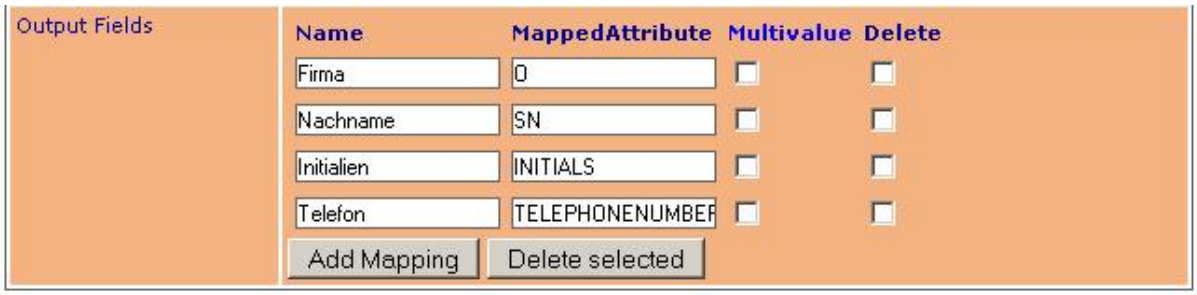
Request Konfiguration 3

Wie zu sehen ist, wurde für jedes ausgewählte Attribut eine neue Zeile in der Tabelle "Output Fields" angelegt. Jede dieser Zeilen repräsentiert genau ein Ausgabefeld, das später in einen Container eingebunden werden kann.

Die Bedeutung der einzelnen Spalten ist dabei wie folgt:

- **Name** - Der Name unter dem das Ausgabefeld später verfügbar sein wird.
- **Mapped Attribute** - Der Name des Attributes, das dieses Feld liefert.
- **Multivalue** - Ist diese Checkbox aktiviert, so wir ein Attribut mit mehreren Werten erwartet. Diese wird als einspaltige Tabelle zurückgeliefert. *Ist diese Option deaktiviert, so wird im Falle eines Attributes mit mehreren Werten der erste Wert eingesetzt.*
- **Delete** - Diese Spalte dient zum entfernen eines Feldes. Mehrere können auf diese Weise ausgewählt werden und dann mit Hilfe des Buttons "Delete selected" gesammelt gelöscht werden.

Als Name der Ausgabefelder wird standardmässig der Name des Attributes angenommen. Dieser Name kann in der Spalte "Name" editiert werden:



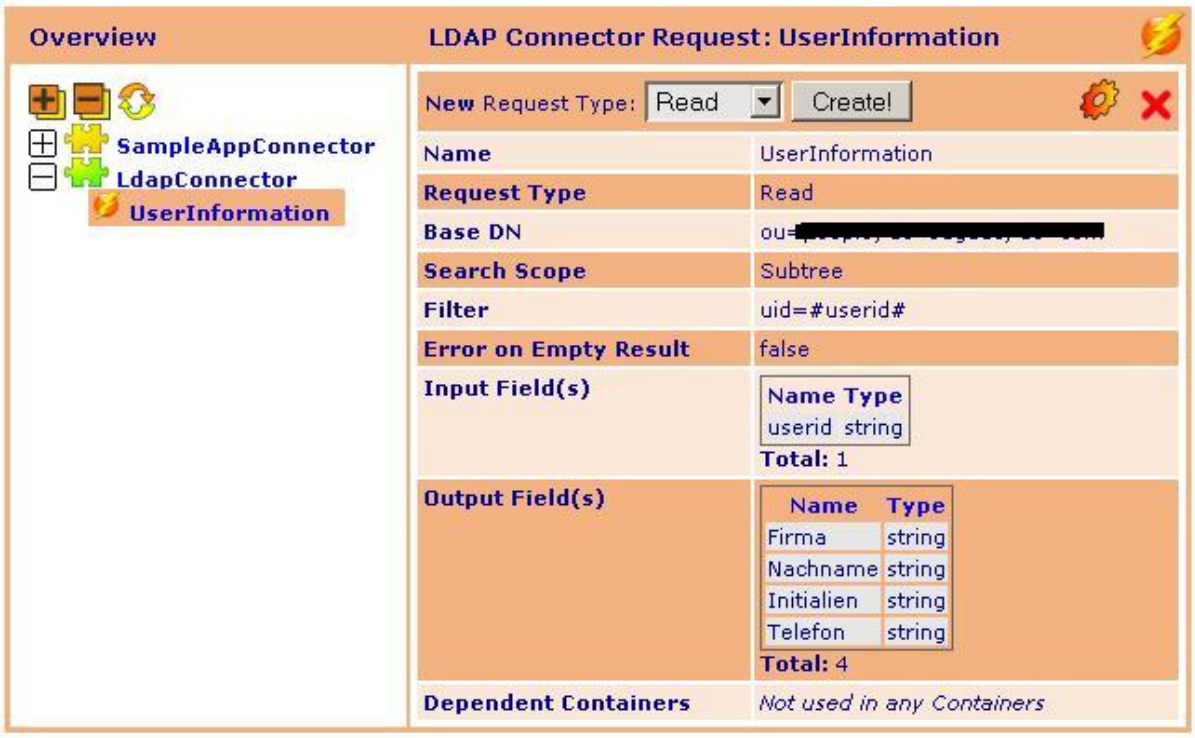
Request Konfiguration 3

Über den Button **"Add Mapping"** können weitere Ausgabefelder hinzugefügt werden. Dies ist notwendeig, wenn z.B. ein Attribut nicht in den Beispieldaten auftaucht, bei anderen

Objekten aber vorhanden ist. Auf diesem Wege kann auch ein Request definiert werden, ohne zum Erstellungszeitpunkt Zugriff auf das Verzeichnis zu haben.

3.5. Speichern des Requests

Ist die Konfiguration abgeschlossen, kann der Request über den Button "Save" gespeichert werden. Anschließend wird erneut die Request-Übersicht präsentiert, diesmal ist der neu erstellte Request in der Tabelle aufgeführt:



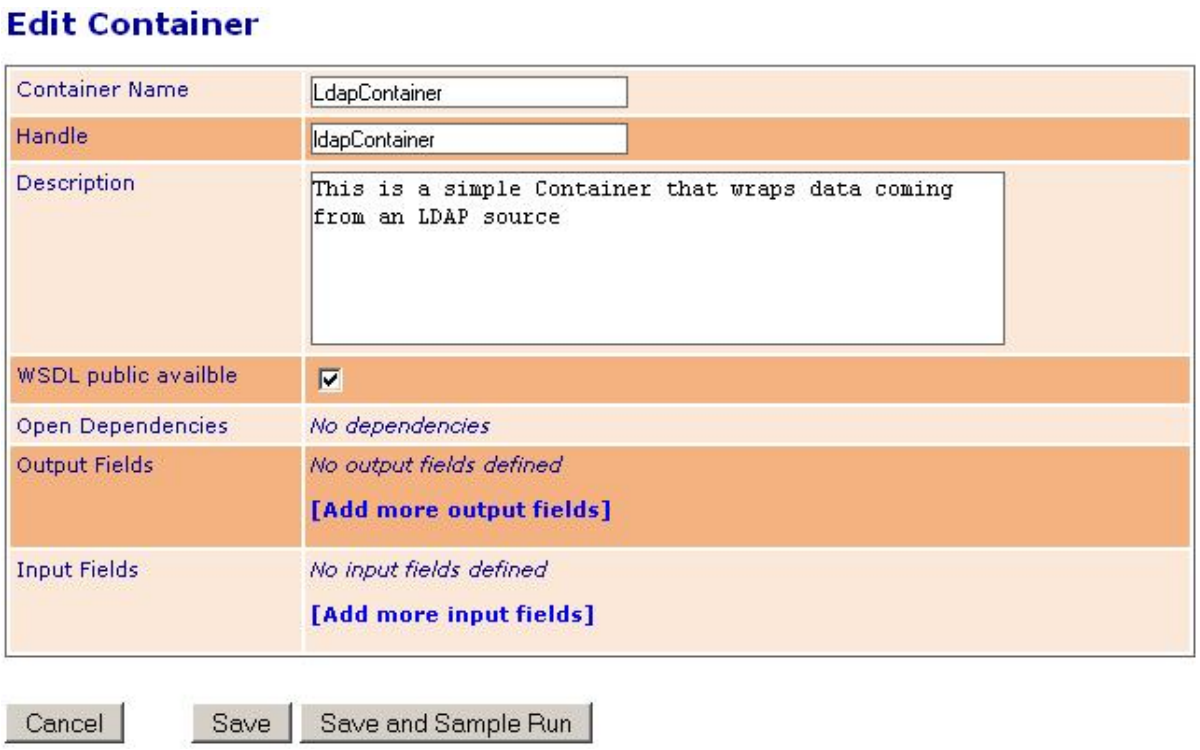
Request Konfiguration 3

Ab diesem Zeitpunkt steht der Request für die Nutzung innerhalb eines Containers bereit.

4. Erstellen des Containers

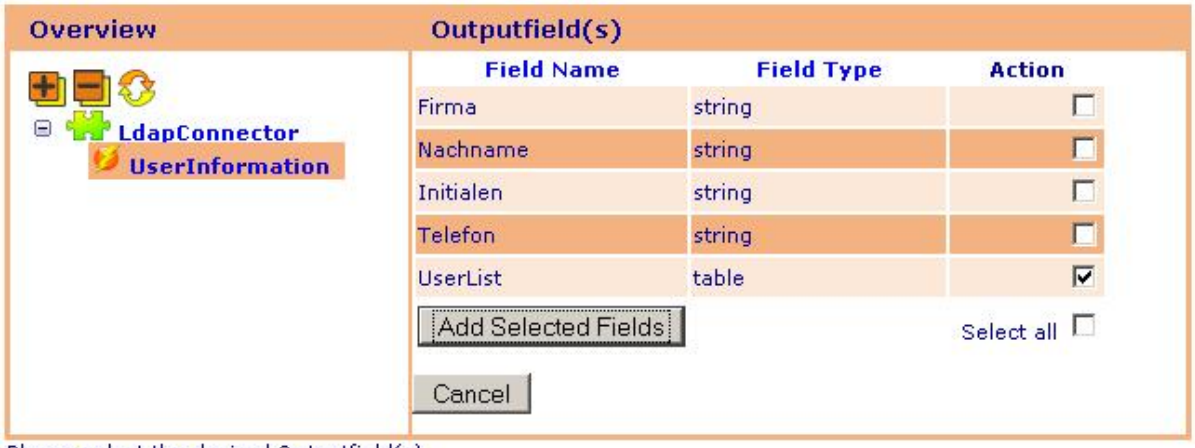
Als nächstes wird ein Container erstellt, der den eben erstellten Request nutzt. Im Menü muss hierzu der Menüpunkt "Containers" und anschließend "Create Container" ausgewählt werden.

Im ersten Schritt sind die Daten entsprechend der Abbildung einzutragen:



xxx

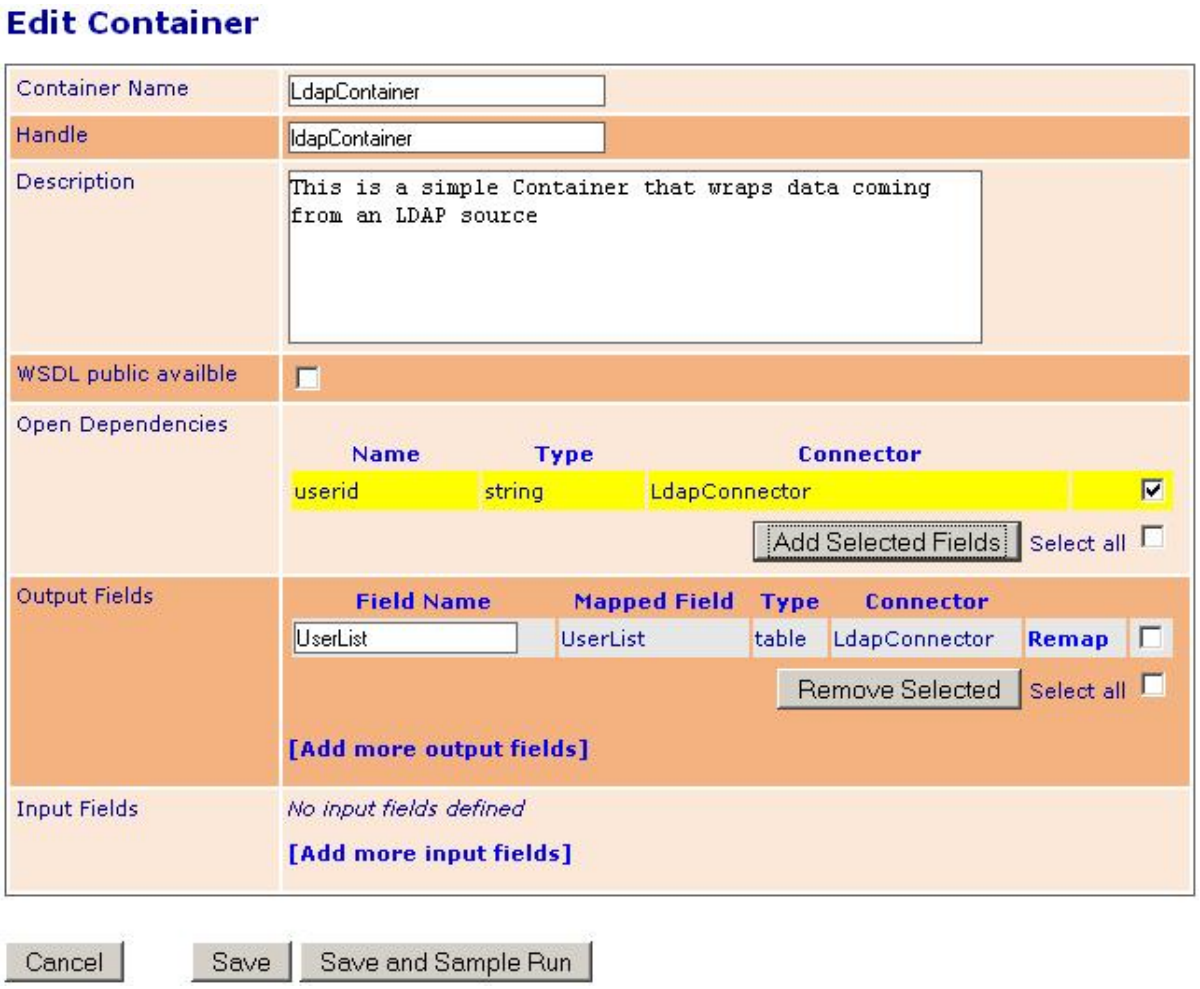
Ein Klick auf den Link "Add more output fields" führt zur Page mit den Connectoren in der Baumstruktur. Dort wird nun der soeben erstellte Connector mit dem erstellten Request ausgewählt. Eine Übersicht über alle verfügbaren Output Fields wird rechts angezeigt. Dort wird die Tabelle "UserList" selektiet:



Please select the desired Outputfield(s)

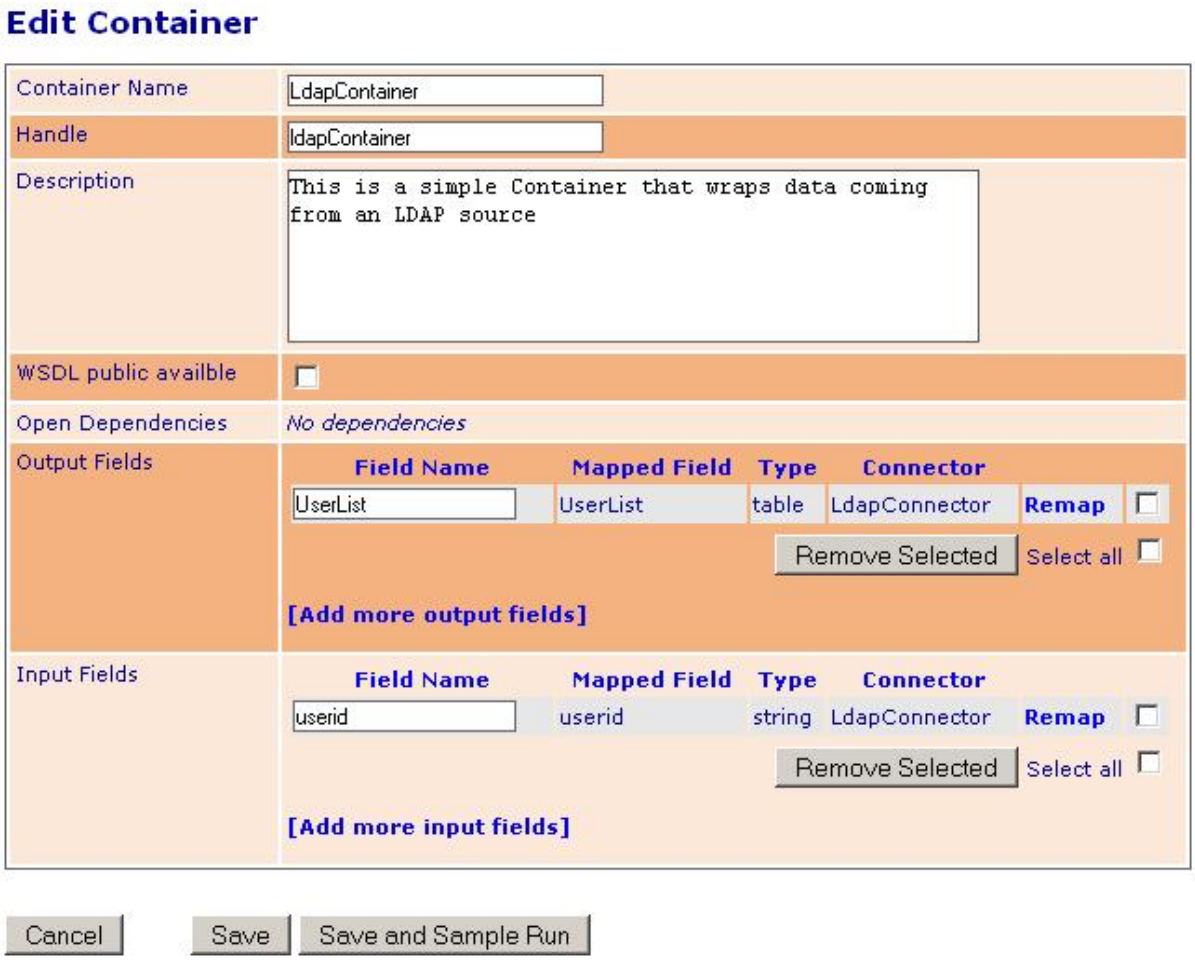
xxx

Anschließend sollte die Konfiguration wie folgt aussehen:



xxx

Als nächstes muss das Eingabefeld "userid" dem Container hinzugefügt werden. Dazu wird das Eingabefeld im Abschnitt "Open Dependencies" selektiert und mit "Add Selected Fields" dem Container hinzugefügt. Die Konfiguration des Containers sollte jetzt so aussehen:



xxx

Ein Klick auf "Save and Sample Run" speichert den Container und zeigt sofort die Sample Run Page an, in der der Container getestet werden kann:

xxx

Das Eingabefeld "userid" wird als Textfeld angezeigt, in dem der Wert eingegeben werden muss, der in den Request einfließt. Hier kann nun also der Filter spezifiziert werden, der bei der Suche auf das Attribut "uid" angewendet wurde. Wird Hier z.B. eine Wildcard suche mit Hilfe des Zeiches "*" durchgeführt, so werden alle gefundenen Objekte in einer Tabellenstruktur zurückgeliefert:

Nach einem Klick auf "Fire!" wird das Ergebnis präsentiert:

xxx

Wurde die Checkbox "Show XML Request/Response" aktiviert, so sind auch die XML-Strukturen für diesen Container zu sehen:

XML Request

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<ContainerRunRequest>
  <container handle="ldapContainer" id="51314689">LdapContainer</container>
  <token>0d26eb2f63fcade389302a0a4164d107</token>
  <username>admin</username>
  <password>MD5 Hash of you password</password>
  <data type="field" varType="string" id="51347457" dataItemName="userid">
    <value>c*</value>
  </data>
</ContainerRunRequest>
```

XML Response

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<ContainerRunResponse>
  <returnCode>0</returnCode>
  <containerRunMessage>Container was run. See results for more details</containerRunMessag
  <ConnectorErrors></ConnectorErrors>
  <data type="table" id="51347458" dataItemName="UserList" colCount="4" rowCount="3">
    <columns>
      <column name="Firma" x="0"></column>
      <column name="Nachname" x="1"></column>
      <column name="Initialen" x="2"></column>
      <column name="Telefon" x="3"></column>
    </columns>
    <value x="0" y="0">SAGA D.C. GmbH</value>
    <value x="0" y="1">SAGA D.C. GmbH</value>
    <value x="0" y="2"></value>
    <value x="1" y="0">Grotepass</value>
    <value x="1" y="1">Spaeth</value>
    <value x="1" y="2">Claudia</value>
    <value x="2" y="0">CGr</value>
    <value x="2" y="1">CSp</value>
    <value x="2" y="2"></value>
    <value x="3" y="0">+49 6731 9428 12</value>
    <value x="3" y="1">+49 6731 9428 15</value>
    <value x="3" y="2"></value>
  </data>
</ContainerRunResponse>
```

xxx

Der Container ist nun erstellt und getestet und kann über die verschieden Schnittstellen von Galaxy abgefragt werden.