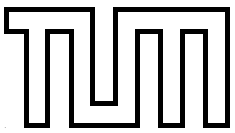


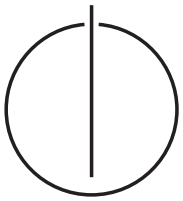
Technische Universität München

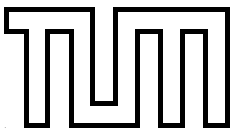
Fakultät für Informatik

Bachelorarbeit in Informatik

Integration eines Single-Sign-On-Verfahrens in ein Java-basiertes Web Kollaborationswerkzeug

Jakob Class





Technische Universität München

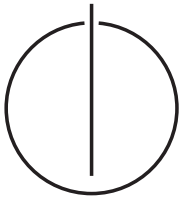
Fakultät für Informatik

Bachelorarbeit in Informatik

Integration eines Single-Sign-On-Verfahrens in ein Java-basiertes Web Kollaborationswerkzeug

Integration of a Single-Sign-On-Service into a Java-based Web Collaboration Tool

Bearbeiter:	Jakob Class
Aufgabensteller:	Prof. Dr. Florian Matthes
Betreuer:	Christian Neubert
Abgabedatum:	15. November 2009



Ich versichere, dass ich diese Bachelorarbeit selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 15. November 2009

. . . . . .
(Unterschrift des Kandidaten)

Abstract

Viele IT-Ressourcen werden in öffentlichen Umgebungen vor unberechtigtem Zugriff durch geeignete Authentifizierungsverfahren geschützt. Durch die Vielzahl verteilter Ressourcen müssen die Anwender oft mehrere unterschiedliche Zugangsdaten pflegen. Verschiedene Single-Sign-On Ansätze bieten dafür eine adäquate Problemlösung.

In dieser Arbeit wird OpenID, ein freier Single-Sign-On Standard für Webumgebungen, analysiert und in das Thema Identity Management eingeordnet. Neben einer Beschreibung des freien Standards wird insbesondere gezeigt, welche Gefahren und Schwachstellen der Einsatz von OpenID bergen kann und welche mögliche Gegenmaßnahmen getroffen werden können.

Die Integration von OpenID in das Tricia Web-Framework ist Gegenstand des praktischen Teils dieser Arbeit. Nach der Analyse des Web-Frameworks folgt die vierphasige Entwicklung eines Plugins, das den Betrieb von Tricia mit der Unterstützung von OpenID ermöglicht.

In der ersten Phase der Entwicklung werden die Anforderungen ermittelt und die erforderlichen Anwendungsfälle identifiziert. Anhand einer Analyse dieser Anforderungen und Anwendungsfälle wird in der zweiten Phase ein Modell entwickelt, das die Anwendungsdomäne beschreibt. Aufbauend auf diesem Modell wird in der dritten Phase die Architektur des Plugins entworfen und eine OpenID Bibliothek zur Bereitstellung der OpenID Kernfunktionalitäten ausgewählt. Schließlich wird mit der Implementierung die Entwicklung des Plugins in der vierten und letzten Phase abgeschlossen.

Inhaltsverzeichnis

1. Einführung

1.1. Motivation

Identity Management hat sich in den letzten Jahren zu einem wichtigen Themengebieten der Informatik entwickelt. Identity Management bietet die Grundlage für Zugriffskontrollen in der digitalen Welt, die vor allem im Zeitalter der weltweiten Vernetzung durch das Internet immer mehr an Bedeutung gewinnen. Für zahlreiche computerbasierte Dienste wie z.B. Online-Shopping, Online-Banking, Webportale oder Intranetprotale in Firmen benötigen die Benutzer eine digitale Identität, um kontrollierten und geschützten Zugriff auf verschiedene Ressourcen erlangen zu können.

Für die Verwaltung digitaler Identitäten existieren diverse Ansätze. Oft verwalten Dienstanbieter wie zum Beispiel Firmen oder Organisationen die Benutzerkonten ihrer Kunden, Mitarbeiter oder Mitglieder lokal in ihren Systemen. Nimmt ein Benutzer mehrere Dienste bei verschiedenen Anbietern in Anspruch, muss er bei jedem Dienstanbieter ein Konto registrieren um die gewünschten Dienste nutzen zu können. Durch die getrennte und über diverse Anbieter verteilte Datenspeicherung sammeln sich über die Zeit zahlreiche Benutzerkonten an, die letztendlich aber alle genau einer realen Person zugeordnet werden können. Die Fülle an verschiedenen Konten bedeutet einen großen Aufwand für den Benutzer, da dieser sich für jeden Anbieter die zugehörigen Zugangsdaten merken muss und bei einer Änderung seiner persönlichen Daten alle seine Konten separat aktualisieren muss.

Abhilfe würde *eine* digitale Identität für jeden Benutzer schaffen, sodass die persönlichen Daten nur an einer zentralen Stelle zu verwalten wären und alle digitalen Aktivitäten mit dieser Identität ausgeübt werden könnten, egal bei welchem Anbieter.

Es existieren schon zahlreiche Ansätze, die versuchen, dieser einen digitalen Identität näher zu kommen. So werden zum Beispiel in vielen Firmen sogenannte Single-Sign-On (SSO) Systeme eingesetzt, mit dessen Hilfe die Mitarbeiter über ein Benutzerkonto alle firmeninternen Anwendungen und Dienste wie zum Beispiel das Betriebssystem, den Mail-Server oder die Groupware nutzen können. Es existieren sogar Ansätze dieser Art über Firmengrenzen hinweg: hier werden Benutzerdaten über große Netzwerke wie das Internet ausgetauscht. Ein Beispiel dafür ist der offener Standard *OpenID*, mit dem SSO Umgebungen im Web realisiert werden können.

In dieser Arbeit wird der OpenID Standard näher betrachtet und in das Themengebiet des Identity Managements eingeordnet. Der praktische Einsatz wird anhand einer Integration von OpenID in ein vorhandenes Web Collaboration Tool des Lehrstuhl für Software Engineering for Business Information Systems (sebis) an der Technischen Universität München illustriert.

1.2. Aufgaben und Ziele

Es sollen Einsatzmöglichkeiten und Grenzen sowie die technische Funktionsweise von OpenID analysiert und beschrieben werden.

Ziel der Arbeit ist unter Verwendung einer ausgesuchten Referenzimplementierung das am sebis-Lehrstuhl entwickelte Tricia Web-Framework (siehe Kapitel 8) OpenID fähig zu machen, das heißt OpenIDs von anderen Providern zur Authentifizierung zu nutzen und die Tricia-Benutzerkennungen zum Login auf anderen Plattformen bereitzustellen.

1.3. Überblick über den Aufbau der Arbeit

Die Arbeit gliedert sich in drei Teile. In Teil I wird grundlegendes Wissen zum Thema Identity Management und insbesondere zu Single-Sign-On vermittelt.

Im zweiten Teil wird der freie Web Single-Sign-On Standard OpenID vorgestellt. Nach einführenden Beispielszenarios, die die Funktion von OpenID aus Sicht des Benutzers beschreiben, wird in Abschnitt 5 die Technik des Standards erklärt. Im letzten Abschnitt von Teil II werden potentielle Schwachstellen und Gefahren von OpenID erörtert und mögliche Gegenmaßnahmen vorgestellt.

In Teil III wird die Entwicklung eines Plugins dokumentiert, mit dem OpenID in Tricia integriert wird. Nach einer Beschreibung von Tricia in Kapitel 8 folgt in Kapitel 9 die Ermittlung der Anforderungen an das Plugin. Basierend auf der Analyse dieser Anforderungen in Kapitel 10, wird in Kapitel 11 die Architektur des Plugins entworfen, dessen Implementierung schließlich in Kapitel 12 dokumentiert wird.

Abschließend findet in Kapitel 13 eine Analyse und Bewertung der Ergebnisse dieser Arbeit statt.

Teil I.

Identity Management: Theorie und Grundlagen

2. Zugangs- und Zugriffskontrolle

2.1. Digitale Identität

Eine digitale Identität repräsentiert ein reales Subjekt in einem konkreten IT-System. Bei dem Subjekt muss es sich nicht zwangsläufig um eine Person handeln, auch Geräte oder Programme können das Subjekt einer digitalen Identität darstellen. Eine digitale Identität setzt sich aus Daten beziehungsweise Attributen zusammen, die das Subjekt beschreiben. Die digitale Identität eines Studenten könnte sich zum Beispiel aus den Attributen Vorname, Nachname, und Matrikelnnummer zusammensetzten.

Da für die Repräsentation von Subjekten in jedem IT-System die Führung digitaler Identitäten notwendig ist und diese Systeme in der Regel unabhängig voneinander verwaltet werden, kann ein reales Subjekt mehrere digitale Identitäten besitzen. Die Ausprägung dieser digitalen Identitäten kann je nachdem, welche Daten des Subjekts für das zugrunde liegende System relevant sind, unterschiedlich ausfallen.

Oft kann ein Subjekt sogar verschiedene Rollen innerhalb eines Systems einnehmen. Je nach ausgeprägter Rolle können die Attribute der digitalen Identität variieren. Beispielsweise kann eine Person bei einem Online-Shop Einkäufe als Privat- oder Geschäftskunde tätigen. In beiden Fällen agiert die gleiche reale Person mit dem System des Online-Shops, allerdings unterscheiden sich je nach eingenommener Rolle bestimmte Attribute der digitalen Identität, wie zum Beispiel die Telefonnummer oder die Adresse des Kunden.

2.2. Authentisierung und Authentifizierung

Um den Bezug zwischen einem realen Subjekt und einer digitalen Identität in einem IT-System herstellen zu können ist es erforderlich, dass das Subjekt dem System seine Identität nachweist. Für den Nachweis und die Überprüfung einer Identität existiert im Englischen ein Begriff, der diese beiden Vorgänge in einem Wort beschreibt: *authentication*. Die deutsche Sprache stellt hingegen zwei Begriffe zur Verfügung, die relativ häufig als Synonyme verwendet werden, obwohl deren Bedeutungen zwar ähnlich, aber nicht identisch sind [MA08, S. 54]:

- **Authentisierung**: Vorgang des Nachweises der eigenen Identität.
- **Authentifizierung**: Vorgang zur Überprüfung einer behaupteten Identität.

Für den Nachweis beziehungsweise die Überprüfung einer Identität existieren verschiedene Möglichkeiten, sogenannte *Authentication Factors* [Win05, S. 51]. Diese lassen sich in drei Kategorien einteilen, je nachdem wie der Benutzer seine Identität nachweist [Reh08, S. 47]:

- **Wissensbasiert**: Informationen, die nur der Benutzer kennt wie zum Beispiel ein Passwort.

- **Besitzbasiert**: Dinge, die nur der Benutzer besitzt wie zum Beispiel eine Smartcard oder ein digitales Zertifikat.

- **Eigenschaftsbasiert**: Merkmale, die den Benutzer eindeutig identifizieren wie zum Beispiel ein Fingerabdruck.

Oft wird nicht nur ein Verfahren zur Authentisierung/Authentifizierung verwendet, sondern eine Kombination aus verschiedenen Verfahren. Im Allgemeinen gilt, dass die Sicherheit steigt, je mehr der oben genannten Kategorien abgedeckt werden. Gleichzeitig steigt aber auch der Aufwand für die Betreiber und die Nutzer beim Einsatz mehrerer Verfahren, da zusätzliche Daten erhoben und gespeichert werden müssen.

Im Folgenden werden gängige Verfahren für den Nachweis beziehungsweise die Überprüfung einer Identität vorgestellt [Win05, S. 51-58].

2.2.1. Benutzerkennung und Passwort

Bei diesem Verfahren erfolgt die Anmeldung durch die Angabe einer Benutzerkennung und eines Passworts. Durch die Prüfung auf Übereinstimmung der Benutzerangaben kann die Echtheit der behaupteten Identität festgestellt werden. Der Nachteil einer passwortbasierten Anmeldung ist, dass die Sicherheit unmittelbar mit der Zusammensetzung der Passwörter einhergeht: Leicht zu merkende Passwörter sind in der Regel auch leicht zu erraten, wohingegen Passwörter mit einem hohen Sicherheitsgrad in der Regel sehr kompliziert aufgebaut sind. Dies hat zur Folge, dass unsichere Passwörter verwenden werden, Passwörter notiert und damit für andere sichtbar gemacht werden oder Passwörter oft vergessen werden. Trotzdem kommt dieses Verfahren in vielen System zum Einsatz, da es sowohl für die Betreiber als auch für die Nutzer relativ einfach zu realisieren beziehungsweise zu bedienen ist.

2.2.2. Challenge-Response Systeme

Bei Challenge-Response Systemen wird dem Benutzer bei der Anmeldung eine Aufgabe (engl. *Challenge*) gestellt. Die korrekte Antwort (engl. *Response*) des Problems sollte nur dem Besitzer der zugrunde liegenden Identität bekannt sein. Die im vorherigen Abschnitt beschriebene passwortbasierte Anmeldung kann eigentlich auch als Challenge-Response System betrachtet werden: Das Problem ist dabei die Frage nach dem Passwort zur angegebenen Benutzerkennung, die Lösung das Passwort selbst. Allerdings handelt es sich dabei um eine sehr vereinfachte Variante des Challenge-Response Verfahren. Ein weitaus höherer Sicherheitsgrad kann erreicht werden, wenn dem Benutzer nicht immer die gleiche Frage gestellt und somit auch nicht immer die gleiche Antwort gefordert wird. Anstelle einer fest vorgegebenen Lösung ist dem Benutzer dann nur ein Algorithmus bekannt mit dem er die Lösungen für eine bestimmte Menge von Aufgaben generieren kann. So können dem Benutzer zufällig ausgewählte Aufgaben gestellt werden, so dass die richtigen Antworten bei verschiedenen Anmeldungen variieren. Ein „Abhören“ der richtigen Lösung, wie sie bei passwortgeschützten Systemen möglich ist, wird dadurch erschwert. Ein sehr simpler Algorithmus wäre beispielsweise die Addition mit 10: Das System übergibt dem Benutzer eine Zufallszahl, die Summe aus 10 und dieser Zufallszahl ist dann die Lösung der Aufgabe.

2.2.3. Biometrische Verfahren

Biometrische Authentifizierungssysteme nutzen bestimmte biometrische Eigenschaften, wie zum Beispiel Fingerabdrücke oder Irisscans, um den Benutzer zu identifizieren. Der Vorteil solcher Verfahren liegt darin, dass die Identifikationsmerkmale in der Regel fest mit dem Benutzer verbunden sind und somit nur bedingt von Dritten missbraucht werden können. Allerdings sind biometrische Authentifizierungssysteme in der Regel sehr kostspielig und nicht sonderlich skalierbar, da sie eine Anschaffung und Installation spezieller Eingabe- und Scangeräte voraussetzten.

2.2.4. Smart Cards

Smart Cards sind Kreditkarten ähnliche Karten, die mit einem Magnetstreifen oder einem eingebauten Chip ausgestattet sind, auf denen beliebige Informationen gespeichert werden können. Ähnlich wie bei den biometrischen Verfahren sind auch bei diesen Authentifizierungssystemen Lesegeräte für die Smart Cards erforderlich.

In der Praxis, wie zum Beispiel bei Bankautomaten, werden Smart Cards oft in Kombination mit einem PIN eingesetzt.

2.3. Autorisierung

Die Aufgabe der Authentisierung und der Authentifizierung ist die Feststellung der Identität eines Benutzers. Die anschließende Prüfung, ob und auf welche Ressourcen der identifizierte Benutzer zugreifen darf, fällt unter den Begriff der Autorisierung. Grundsätzlich existieren drei verschiedene Strategien zur Verwaltung von Zugriffsrechten [DH08, S. 242f]:

- **Discretionary Access Control (DAC)**: DAC ist ein einfaches Zugriffsmodell, bei dem die Vergabe von Zugriffsrechten auf Ressourcen durch die Erzeuger erfolgt. Dieses Modell hat allerdings einige Schwachstellen [KE06, S. 339], da der Erzeuger gleichzeitig auch als Eigner einer Ressource betrachtet wird. So sind zum Beispiel Daten die von einem Angestellten erzeugt wurden in der Regel Eigentum der Firma, für die der Angestellte tätig ist. Trotzdem kann der Angestellte frei entscheiden, wer Zugriff auf diese Daten haben kann.

- **Mandatory Access Control (MAC)**: Das Problem, dass Daten vom Erzeuger potentiell an Unbefugte weitergegeben werden können, wird bei MAC durch die Einführung von hierarchischen Sicherheitsstufen verhindert. Jedem Benutzer und jeder Ressource wird dazu eine Sicherheitsstufe zugeordnet. Beim Erstellen einer Ressource wird der Ressource automatisch die Sicherheitsstufe des Erzeugers zugewiesen. Gleichzeitig muss ein Benutzer mindestens der Stufe zugeordnet sein, die einer Ressource zugewiesen ist, auf die er zugreifen möchte.

- **Role Based Access Control (RBAC)**: Bei der rollenbasierten Zugriffskontrolle werden Zugriffsrechte nicht einzelnen Benutzern sondern bestimmten Rollen zugewiesen, die von den Benutzern eingenommen werden können. Der Vorteil dieses Modell liegt darin, dass die Komplexität der Verwaltung von Zugriffsrechten durch die Zusammenfassung mehrerer Benutzer reduziert werden kann.

3. Single-Sign-On

Single-Sign-On (SSO) ist ein Mechanismus zur Einmalanmeldung. In SSO Systemen werden die Benutzer zu Beginn einer Sitzung einmalig authentifiziert. Anschließend können die Benutzer innerhalb des Systems auf alle Dienste und Ressourcen zugreifen, für die sie autorisiert sind, ohne sich erneut authentisieren zu müssen.

3.1. Vor- und Nachteile

SSO Systeme bieten für verschiedenen Interessengruppen diverse Vorteile:

- *Sicherheitsgewinn*: Da sich die Nutzer nur ein Passwort merken müssen, können komplexere Passwörter gewählt werden. Das Risiko durch Abfangen von Zugangsdaten wird gemindert, da diese Daten nur einmal zu Beginn der Sitzung übertragen werden müssen. Zusätzlich begünstigt eine zentrale Anmeldung den Einsatz verschiedener Authentication Factors (siehe Abschnitt 2.2).

- *Vereinfachte Administration*: In SSO Systemen können die digitalen Identitäten zentral verwaltet und müssen nicht in jedem Subsystem gepflegt werden.

- *Erhöhter Benutzerkomfort*: Die Benutzer müssen sich nur ein Passwort merken. Zusätzlich entfallen mit SSO sich wiederholende Anmeldeprozeduren.

Allerdings birgt der Einsatz von SSO Mechanismen die Gefahr eines *Single Point of Failure*:

- *Verfügbarkeit*: Fällt der zentrale Anmeldungsdienst aus, ist der Zugriff auf alle Dienste und Ressourcen der SSO Umgebung nicht mehr möglich.

- *Sicherheit*: Gelangt ein Unberechtigter an die Zugangsdaten eines autorisierten Benutzers oder kompromittiert den zentralen Anmeldemechanismus, hat er vollen Zugriff auf die einzelnen Komponenten des SSO Systems.

3.2. Identity Federation

SSO Systeme werden in der Regel im Rahmen einer Organisation oder einer Firma realisiert. Die zunehmenden Einbindung der IT in Geschäftsprozesse und die weltweite Vernetzung im Zuge der Globalisierung führt immer häufiger zu einem digitalen Zusammenschluss mehrere autonomer Firmen und Organisationen zu einer sogenannten *Identity Federation* [MA08, S. 72f].

Das Ziel solcher Zusammenschlüsse ist die Schaffung sogenannter *Federated Identities* [Win05, S. 118f]. Eine Federated Identity ist eine digitale Identität, die im Kontext einer Identity Federation existiert und dort dezentral verwaltet wird. Im Gegensatz zu einer zentral verwalteten digitalen Identität stammen die Attribute einer föderierten Identität aus

verschiedenen Quellen innerhalb einer Federation. Durch den Informationsaustausch zwischen den Mitgliedern innerhalb einer Identity Federation können diese Attribute bei Bedarf zusammengeführt werden.

Der Vorteil dieses Ansatzes ist, dass eine Vielzahl von Datenquellen mit jeweils unterschiedlichen Informationen existieren können und keine große zentrale Datenbank benötigt wird, in der alle Daten gespeichert werden. Die verschiedenen Informationen können dann an den Stellen gepflegt werden, an denen sie auch wirklich „entstehen“.

Eine Identity Federation baut auf das Vertrauen unter ihren Mitgliedern auf. Da die Identitätsdaten dezentral verwaltet und zwischen den Teilnehmern einer Federation untereinander ausgetauscht werden, müssen sie sich auf die Korrektheit der Daten anderer Teilnehmer verlassen können.

3.2.1. Identity Provider, Service Provider und Relying Party

Die Teilnehmer einer Identity Federation können zwei verschiedene Rollen einnehmen:

Identity Provider Ein Identity Provider (IP) verwaltetet digitale Identitäten und stellt diese für andere Teilnehmer bereit. Er ist in der Lage ein Subjekt, dessen digitale Identitäten er verwaltet, zu authentifizieren.

Service Provider/Relying Party Ein Service Provider (SP) stellt bestimmte Dienste und Ressourcen für autorisierte Benutzer bereit, ohne selbst die digitalen Identitäten dieser Benutzer zu verwalten. Für die Autorisierung der Benutzer fordert ein SP die nötigen Identitätsdaten bei einem IP an. Da der SP diese Identitätsdaten nur sinnvoll nutzen kann, wenn er dem IP vertraut, bezeichnet man einen SP oft auch als Relying Party (RP).

Teil II.

OpenID

4. Einführung und Übersicht

OpenID ist ein offener Standard, der ein SSO System für Webseiten beschreibt. Es handelt sich hierbei um ein Drei-Parteien-System, das sich aus dem Benutzer beziehungsweise dessen Browser (User-Agent), der Relying Party (RP) und dem dem Identity Provider (IP) zusammensetzt [Reh08].

Der IP wird in der OpenID-Terminologie meist als OpenID Provider (OP) bezeichnet und ist für die Verwaltung der Benutzerdaten und die Authentifizierung des Benutzers zuständig. Ein Benutzer wird in einer OpenID Umgebung eindeutig mit einer URL identifiziert. Diese URL stellt die Kennung des Benutzers dar und wird auch als *Identifier* oder *OpenID* bezeichnet. Für die Form der OpenID existieren keinerlei Einschränkungen. Üblicherweise setzt sie sich aus Gründen der Lesbarkeit aus dem Benutzernamen und der Domain des OpenID Providers (OPs) zusammen. So wären zum Beispiel für einen Benutzer *bob*, der bei dem OP *openid-provider.com* registriert ist, folgende OpenIDs denkbar:

- *http://bob.openid-provider.com*
- *http://openid-provider.com/bob*
- *http://openid-provider.com/?user=bob*

Die RP, die auch als *Consumer* bezeichnet wird, ist eine Webapplikation bei der sich der Benutzer mit seiner OpenID anmelden kann, um dort auf Ressourcen zuzugreifen. Die Authentifizierung erfolgt jedoch durch den OP. Zusätzlich erlauben OpenID Protokollerweiterungen den Austausch von Benutzerdaten wie z.B. Name oder Geburtsdatum zwischen RP und OP. Damit steht dem Benutzer ein zentrales Benutzerkonto bei seinem OP zur Verfügung, mit dem er sich bei beliebigen RPs anmelden kann, ohne dort ein neues Konto erstellen zu müssen.

Aufbau des Kapitels In diesem Kapitel soll die Funktionsweise von OpenID aus Sicht des Benutzers mit zwei einfachen Bespielen verdeutlicht werden. Um einen möglichst schnellen Einblick gewähren zu können, werden technische Prozesse, die in Kapitel 5 behandelt werden und im Hintergrund ablaufen, bewusst außen vor gelassen.

Für die folgenden Beispiele wurde bei MyOpenID, einem kostenlosen OP der Firma JanRain [Janb], die OpenID *http://class.myopenid.com* registriert.

4.1. Anmeldung mit einer OpenID

Zur Illustrierung der einfachen Anmeldung bei einer Relying Party (RP) ohne vorherige Registrierung wurde LiveJournal ausgewählt, ein Hoster und Anbieter für Webblogs [Inc]. LiveJournal stellt für Benutzer, die eine OpenID besitzen, unter http://www.livejournal.com/openid/ eine Anmeldemaske zur Verfügung (Abb. 4.1).

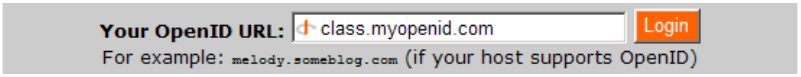


Abbildung 4.1.: Anmeldung bei LiveJournal

Nach der Eingabe seiner OpenID wird der Benutzer per Browser-Redirect zu seinem OP weitergeleitet. Sofern noch keine gültige Session auf dem OP vorhanden ist, muss sich der Benutzer nun bei seinem OP anmelden und eine neue Session starten (Abb. 4.2).

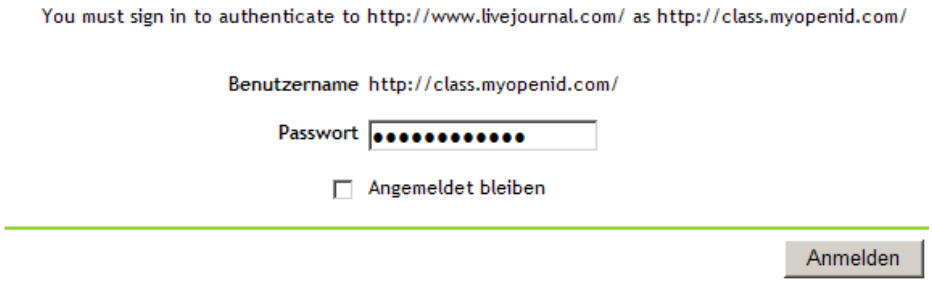


Abbildung 4.2.: Anmeldung bei myOpenID

Erfolgte zuvor noch keine Anmeldung bei der RP, also bei LiveJournal, wird dem Benutzer eine Bestätigungsseite angezeigt, auf der ersichtlich ist, von welcher RP die Authentifizierungsanfrage stammt. Der Benutzer kann an dieser Stelle entscheiden, ob er diese Anfrage zulassen möchte (Abb. 4.3).

Abbildung 4.3.: Bestätigung und Einstellungen für LiveJournal bei myOpenID

Wurde die Authentifizierungsanfrage der RP durch den Benutzer bewilligt, wird dieser dann wieder per Browser-Redirect zur RP (LiveJournal) zurück geleitet und ist dort erfolgreich angemeldet (Abb. 4.4).

4.2. Registrierung mit einer OpenID

Mit LiveJournal wurde im vorherigen Beispiel gezeigt, wie eine Anmeldung ohne vorherige Registrierung abläuft. Oft ist allerdings eine zusätzliche Registrierung bei der RP nötig. Hier kann der Einsatz einer OpenID allerdings auch hilfreich sein: Bei der Registrierung gibt der Benutzer seine OpenID an, dadurch können bestimmte Basisdaten des Benutzers wie z.B. dessen Name oder E-Mail-Adresse von der RP direkt beim OP abgefragt werden und müssen vom Benutzer nicht bei jeder Registrierung erneut angegeben werden. Nach der

Abbildung 4.4.: LiveJournal nach erfolgreicher Anmeldung

Registrierung wird die OpenID mit dem Account des Benutzers in der Datenbasis der RP verknüpft, sodass der Benutzer sich auch in Zukunft mit seiner OpenID bei der RP anmelden kann.

Um eine OpenID basierte Anmeldung zu veranschaulichen wurde als RP PBworks gewählt, ein Hoster für Wikis und andere webbasierte Kollaborations-Applikationen [PBw].

Unter https://my.pbworks.com/?p=openid bietet PBworks eine OpenID-Anmeldemaske an (Abb. 4.5).

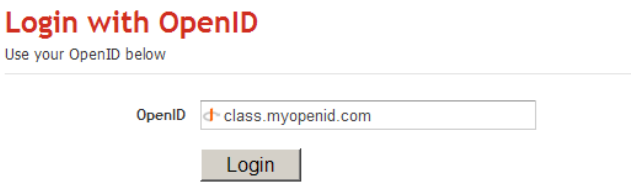


Abbildung 4.5.: Anmeldung bei PBworks

Wie auch bei LiveJournal wird der Benutzer nach der Eingabe seiner OpenID wieder zu seinem OP weitergeleitet. Da mit der Anmeldung im ersten Beispiel bereits eine Session beim OP gestartet wurde, muss der Benutzer sich nicht erneut bei MyOpenID anmelden. Stattdessen bekommt er direkt die Bestätigungsseite für die Anfrage von PBworks zu sehen. Da PBworks neben einer Authentifizierung auch noch Benutzerdaten anfordert, kann der Benutzer zusätzlich auswählen, welche Daten an die RP, also an PBworks, gesendet werden sollen. Bei MyOpenID wird diese Datenweitergabe über sogenannte Rollen gesteuert. Jeder Benutzer kann bei MyOpenID verschiedene Rollen für sich anlegen. Für jede dieser Rollen können diverse Attribute, wie z.B: ein Spitzname, das Alter, ein Benutzerfoto etc. gesetzt werden. Auf der Bestätigungsseite kann der Benutzer dann über die Auswahl einer Rolle entscheiden, welche Daten an die anfordernde RP gesendet werden sollen (Abb. 4.6).

Nach der Bestätigung der Anfrage durch den Benutzer wird dieser wieder zurück zu PBworks geleitet, wo ihm ein Registrierungsformular angezeigt wird, das bereits mit den vom OP übermittelten Benutzerdaten vorbelegt ist (Abb. 4.7).

4.3. Zusammenfassung der Szenarios

In Abbildung 4.8 sind die einzelnen Schritte der beiden Beispielszenarios aus den vorherigen Abschnitten grafisch zusammengefasst:

Abbildung 4.6.: Bestätigung und Einstellungen für PBworks bei myOpenID

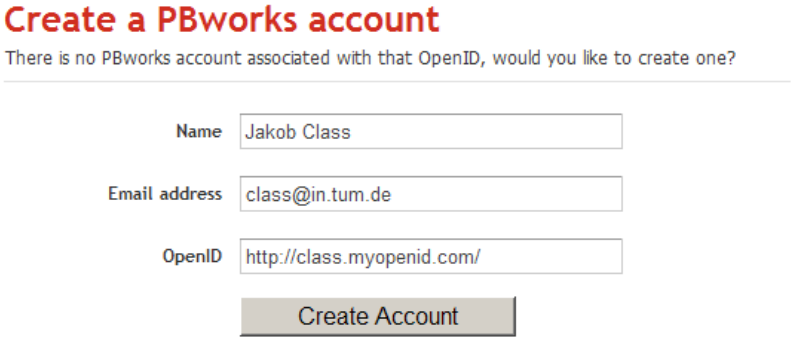


Abbildung 4.7.: Erstellen eines Accounts bei PBworks mit den von myOpenID gelieferten Daten

1. Der Benutzer übergibt LiveJournal (der RP) seine OpenID *http://class.myopenid.com*.
2. LiveJournal ermittelt anhand der übergebenen OpenID den OP des Benutzers und leitet den Benutzer zu myOpenID.
3. Der Benutzer meldet sich bei myOpenID an.
4. Der Benutzer bestätigt bei myOpenID, dass er für LiveJournal authentifiziert werden möchte.
5. MyOpenID leitet den Benutzer mit der Bestätigung, dass dieser authentifiziert wurde zurück zu LiveJournal. Dort ist der Benutzer nun angemeldet und hat eine gültige Sitzung.
6. Der Benutzer übergibt PBworks (der RP) seine OpenID *http://class.myopenid.com*.
7. PBworks ermittelt anhand der übergebenen OpenID den OP des Benutzers und leitet den Benutzer zu myOpenID.
8. Der Benutzer bestätigt bei myOpenID, dass er für PBworks authentifiziert werden möchte und gibt an, welche Benutzerdaten myOpenID an PBworks weiterleiten soll.
9. MyOpenID leitet den Benutzer mit der Bestätigung, dass dieser authentifiziert wurde und den ausgewählten Benutzerdaten zurück zu PBworks. Dort kann der Benutzer mit dem vorbelegten Registrierungsformular seinen persönlichen Account bei PBworks erstellen.

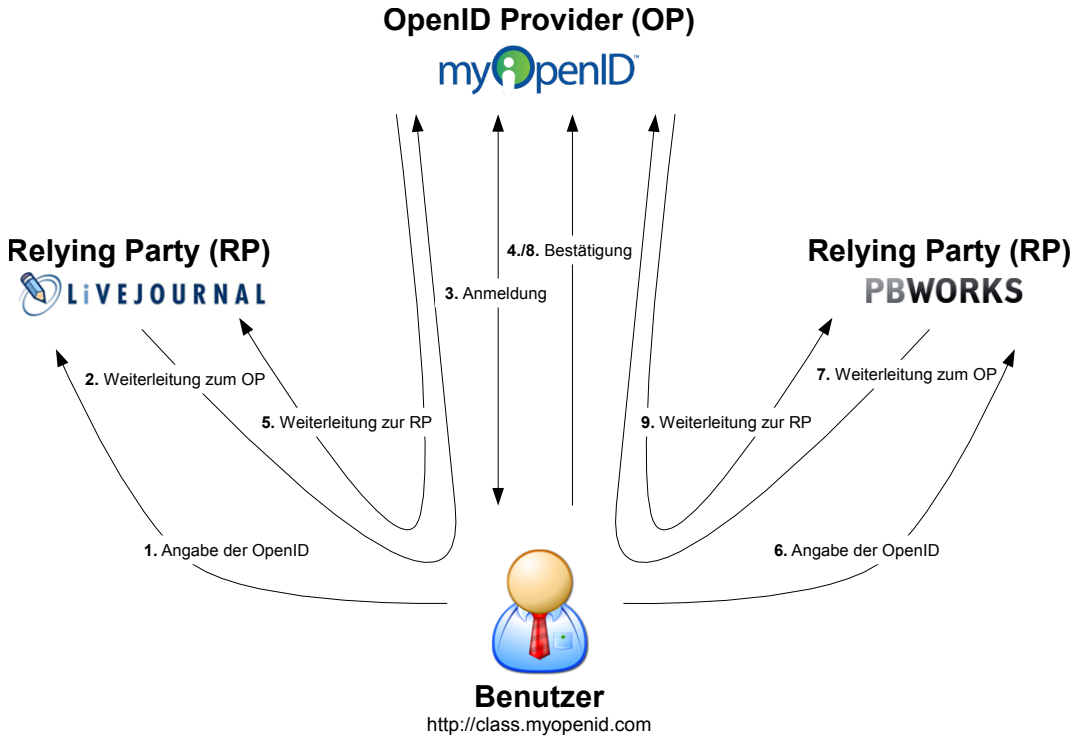


Abbildung 4.8.: Überischt und Zusammenfassung der Schritte aus den Beispielszenarios

5. Spezifikationen

Aufbau des Kapitels Im vorherigen Kapitel wurden die Abläufe einer OpenID Anmeldung aus Sicht des Benutzers beschrieben, um einen groben Überblick über die Funktionsweise des OpenID Protokolls zu gewähren. In diesem Kapitel werden die tatsächlichen technischen Vorgänge behandelt, die während eines OpenID Anmeldeprozesses ablaufen und für den Benutzer teilweise nicht sichtbar sind.

Die OpenID Foundation stellt unter http://openid.net/developers/specs/ eine Reihe von technischen Spezifikationen zur Verfügung, von denen einige in den folgenden Abschnitten näher beschrieben werden:

- OpenID Authentication 2.0 [Ope07]
- OpenID Simple Registration Extension 1.0 [HDJ+06]
- OpenID Attribute Exchange 1.0 [HBS+07]

In Abschnitt 5.1 wird die *OpenID Authentication* beschrieben, die den Kern des OpenID-Protokolls darstellt. In Abschnitt 5.2 wird auf die Protokollerweiterungen *OpenID Simple Registration Extension* und *OpenID Attribute Exchange* eingegangen, die die *OpenID Authentication* ergänzen.

5.1. OpenID Authentication

In diesem Abschnitt werden zentrale Elemente der OpenID Authentication 2.0 Spezifikation vorgestellt. Sie definiert die Eigenschaften und Mechanismen des OpenID Protokolls und legt fest, wie ein OpenID Anmeldeprozess abläuft.

5.1.1. Kommunikation zwischen den einzelnen Teilnehmern

Wie zu Beginn des Kapitels bereits erwähnt wurde, handelt es sich bei OpenID um ein Drei-Parteien-System, das sich aus dem Benutzer beziehungsweise dessen Browser, dem OP und der RP zusammensetzt. Während eines OpenID Anmeldeprozesses kommunizieren diese Teilnehmer miteinander und tauschen diverse Informationen untereinander aus. Die RP fordert die OpenID des Benutzers an, der den Zugriff auf bestimmte Ressourcen wünscht. Um die Identität des Benutzers feststellen und überprüfen zu können, muss die RP eine Authentifizierungsanfrage an den OP stellen. Der OP authentifiziert den Benutzer, indem dieser zum Beispiel seine Kennung und das zugehörige Passwort angibt. Das Ergebnis der Authentifizierung wird dann vom OP an die RP übermittelt. Der Benutzer interagiert über seinen Browser sowohl mit dem OP, bei dem er sich authentisiert, als auch mit der RP bei der er seine OpenID angibt.

Die Teilnehmern kommunizieren über das Hypertext Transfer Protocol (HTTP) miteinander. Dabei kann die Kommunikation zwischen OP und RP entweder direkt oder indirekt ablaufen:

- Bei der direkten Kommunikation senden der OP und die RP die Informationen mit der HTTP POST Methode über eine direkte Verbindung.

- Bei der indirekten Kommunikation tauschen die beiden Partner ihre Nachrichten über den Browser des Benutzers aus: entweder per HTTP Redirect oder über das Abschicken eines HTML Formulars.

5.1.2. OpenID Protokoll Nachrichten

Der OP und die RP tauschen während eines Anmeldevorgangs verschiedene Nachrichten untereinander aus. Diese Nachrichten werden in Form von HTTP Anfragen an den jeweiligen Kommunikationspartner gesendet und enthalten je nach Nachrichtentyp verschiedene Parameter. Die Parameter werden entweder als GET Variablen innerhalb der URL oder als POST Variablen übertragen und beginnen mit dem Prefix „*openid.*“. Folgende Parameter sind für jeden OpenID Nachrichtentyp erforderlich:

- *openid.ns* gibt an, welche Protokoll Version für diese Nachricht verwendet wird. Bei Nachrichten, die die Spezifiaktion in Version 2.0 erfüllen, muss dieser Parameter mit *http://specs.openid.net/auth/2.0* belegt werden.

- *openid.mode* legt fest, um welchen Nachrichtentyp es sich handelt.

Es existieren vier grundlegende OpenID Nachrichtentypen: *associate*, *checkid_immediate*, *checkid_setup* und *check_authentication*.

Die *associate* Nachrichten dienen dem Aufbau einer sogenannten *Association* zwischen der RP und dem OP. Dabei tauschen die beiden Kommunikationspartner einen gemeinsamen Schlüssel untereinander aus, mit dessen Hilfe sie ausgehende Nachrichten signieren und die Gültigkeit eingehender Nachrichten prüfen können.

Stellt eine RP eine *checkid_immediate* Authentifizierungsanfrage an einen OP, ist dieser aufgefordert die Authentifizierung ohne Interaktion mit dem Benutzer durchzuführen. Ist die Authentifizierung nur mit Hilfe einer Benutzerinteraktion möglich, liefert der OP eine negative Bestätigung und teilt der RP mit, dass eine *checkid_setup* Authentifizierungsanfrage notwendig ist.

Bei einer *checkid_setup* Authentifizierungsanfrage fordert eine RP einen OP auf einen Benutzer zu authentifizieren und dessen Identität zu bestätigen. Falls erforderlich kann der OP bei der Authentifizierung mit dem Benutzer interagieren.

Liegt den Kommunikationspartnern kein gemeinsamer Schlüssel vor, mit dem sie sich von der Echtheit der Nachrichten ihres Kommunikationspartners überzeugen können, kann eine Authentifizierungsbestätigung mithilfe der *check_authentication* Nachrichten verifiziert werden.

5.1.3. Ablauf einer OpenID Transaktion

Die OpenID Authentication 2.0 Spezifikation gliedert den Anmeldevorgang in sieben Hauptschritte, die in Abbildung 5.1 als Sequenzdiagramm grafisch dargestellt sind:

1. Durch die Angabe seiner OpenID-URL bei der RP leitet der Benutzer die *Initialisierung der Anmeldung* ein.

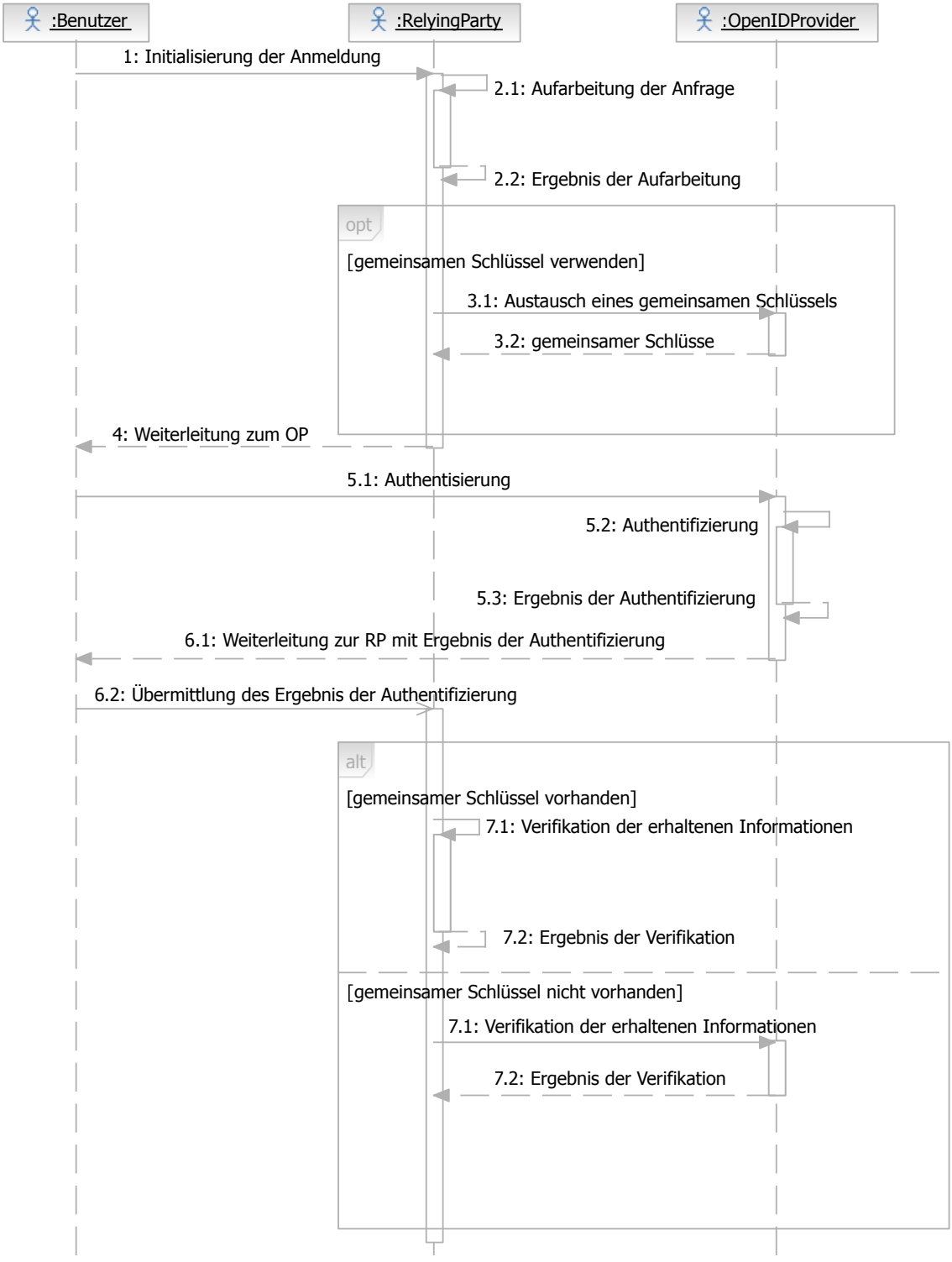


Abbildung 5.1.: Ablauf des OpenID-Anmeldevorgangs

2. Die RP führt eine *Aufarbeitung der Anfrage* durch. Dabei wird die URL zuerst normalisiert, d.h. in ein einheitliches Format umgewandelt. Anschließend werden die für eine Anfrage beim OP nötigen Informationen mit Hilfe dieser URL ermittelt. Neben optionalen Angaben sind das die Version des verwendeten OpenID Protokolls sowie die URL des OP, an die OpenID Authentification Nachrichten gesendet werden können.

3. Optional kann nun mit dem Diffie-Hellman Key Verfahren [Reh08, S. 182ff] der *Austausch eines gemeinsamen Schlüssels* zwischen RP und OP statt finden. Mit diesem Schlüssel kann der OP seine ausgehenden Nachrichten signieren und die RP kann mit Hilfe des Schlüssels die Gültigkeit der empfangenen Nachrichten prüfen. Findet der Schlüsselaustausch nicht satt, muss die RP für jede Überprüfung eine direkte Anfrage an den OP stellen.

4. Anschließend übermittelt die RP dem Browser des Benutzers die Daten für eine *Weiterleitung zum OP*.

5. Der OP überprüft, ob der Benutzer berechtigt ist eine OpenID-Anmeldung durchzuführen und ob er diese für die RP auch wirklich wünscht. Ist dies der Fall, findet die eigentliche *Authentisierung und Authentifizierung* durch den OP statt.

6. Nach der Anmeldung schickt der OP wieder eine *Weiterleitung zur RP mit dem Ergebnis der Authentifizierung* an den Browser des Benutzers.

7. Zum Schluss führt die RP eine *Verifikation der erhaltenen Informationen* durch. Wurde im dritten Schritt ein gemeinsamer Schlüssel ausgetauscht, kann die RP diesen nutzen, um sich von der Echtheit der Informationen zu überzeugen. Ansonsten muss die RP eine direkte Anfrage an den OP stellen, um die Informationen verifizieren zu können.

5.2. Protokollerweiterungen

5.2.1. OpenID Simple Registration Extension

Die OpenID Simple Registration Extension wurde zur Vereinfachung einer OpenID basierten Registrierung entwickelt. Damit ein Benutzer bei der Registrierung seine persönlichen Daten nicht händisch angeben muss, bietet die Protokollerweiterung die Möglichkeit bestimmte Attribute vom OP an die RP zu übertragen, die bei Registrierungen häufig abgefragt werden. Durch den Informationsaustausch zwischen OP und RP wird der Prozess der Registrierung beschleunigt, da die Eingabe dieser Daten durch den Benutzer nicht mehr erforderlich ist. Die Simple Registration Extension ermöglicht den Austausch folgender Attribute in einem vorgeschriebenen Format:

- **nickname**: Eine UTF-8 Zeichenkette für den Spitznamen des Benutzers.
- **email**: Die E-Mail-Adresse des Benutzers.
- **fullname**: Eine UTF-8 Zeichenkette für den vollen Namen des Benutzers.
- **dob**: Eine Zeichenkette mit zehn Buchstaben im Format *YYYY-MM-DD*, die das Geburtsdatum des Benutzers spezifiziert.

- **gender**: Ein Buchstabe, der das Geschlecht des Benutzers bestimmt: „M“ für männlich (engl. *male*), „F“ für weiblich (engl. *female*).
- **postcode**: Eine UTF-8 Zeichenkette für die Postleitzahl des Orts, in dem der Benutzers wohnt.
- **country**: Der nach ISO3166 [ISO] spezifizierte Code für das Land, in dem der Benutzer wohnt.
- **language**: Der nach ISO639 [Con] spezifizierte Code für die Sprache, die der Benutzer bevorzugt.
- **timezone**: ASCII Zeichenkette für die Zeitzonen [Sun], in dem sich der Benutzer aufhält.

5.2.2. OpenID Attribute Exchange

Bei der OpenID Simple Registration Extension ist fest vorgeschrieben, welche Attribute zwischen OP und RP übertragen werden können. Um den Austausch von Identitätsinformationen flexibler gestallten zu können wurde OpenID Attribute Exchange (AX) [HBS+07] entwickelt. Mit dieser Protokollerweiterung können beliebige Identitätsinformationen zwischen den Endpunkten ausgetauscht werden. Für den Austausch dieser Information stellt AX einen Namensraum bereit, in dem eigene Attribute definiert werden können. Bei der Übertragung beziehungsweise Anforderung eines Attributs muss der Typ des Attribut über den sogenannten *type identifier* angegeben werden. Dieser *type identifier* ist eine URL zu einer XML Datei, die das Attribut beschreibt. Solch eine XML Datei könnte für eine E-Mail-Adresse zum Beispiel folgenden Inhalt haben[1]:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:openid="http://axschema.org/type#" xmlns:rdf="http://www.
    w3.org/1999/02/22-rdf-syntax-ns#" xmlns:xhtml="http://www.w3.org
    /1999/xhtml" xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
  <rdf:Description rdf:about="http://axschema.org/contact/email">
    <rdfs:label>Email</rdfs:label>
    <rdfs:comment>Internet SMTP email address as per <xhtml:a href="ftp:
        //ftp.isi.edu/in-notes/rfc2822.txt">RFC2822</xhtml:a></
        rdfs:comment>
    <openid:example>"jsmith@isp.example.com"</openid:example>
    <rdf:type rdf:resource="http://www.w3.org/2001/XMLSchema#
        normalizedString"></rdf:type>
    <openid:regexp>\S+@\S+</openid:regexp>
  </rdf:Description>
</rdf:RDF>
```

Im Gegensatz zur Simple Registration Extension ermöglicht AX einen bidirektionalen Informationsaustausch zwischen OP und RP. Über sogenannte *store* Nachrichten können mit AX auch Attributwerte von der RP an den OP übertragen werden. Dadurch ist es theoretisch möglich, wirklich alle Identitätsinformationen eines Benutzers für alle Anwendungen und Dienste, die er in Anspruch nimmt, zentral bei seinem OP zu speichern.

[1] Quelle: http://axschema.org/

6. Schwachstellen und Gefahren

Aufbau des Kapitels In diesem Kapitel werden einige Aspekte bezüglich der Sicherheit (Abschnitt 6.1 und 6.2) sowie der Usability (Abschnitt 6.3) erörtert, die beim Einsatz des OpenID Standards berücksichtigt werden sollten.

6.1. Phishing

Die größte Gefahr beim Einsatz von OpenID stellen Phishing Attacken dar. Bei diesen Attacken wird die erforderliche Weiterleitung zum OP dazu missbraucht, die Authentifizierungsdaten des Opfers abzufangen. Der Betrüger versucht sein Opfer über eine scheinbar interessante Webseite dazu zu bewegen, sich dort mit seiner OpenID anzumelden. Anstatt das Opfer zu seinem OP weiterzuleiten, leitet die RP des Betrügers den Browser des Opfers zu einen gefälschten OP, der den echten OP des Opfers imitiert. Dieser gefälschte OP sendet die Daten, die das Opfer zur Authentisierung angibt an den Betrüger, der sich anschließend damit beim echten OP anmelden kann. Eine eindrucksvolle Demonstration solcher Attacken bietet [Gmb].

Es existieren einige Lösungsansätze, mit denen solche Phishing Attacken verhindert werden können [Wik]. Allerdings stellen diese Maßnahmen einen Mehraufwand für die Benutzer beziehungsweise die OPs dar. Nachfolgend werden Maßnahmen vorgestellt, mit denen sich die Benutzer vor Phishing Attacken schützen können.

Wachsamkeit: Bei der Authentisierung gegenüber des OP sollte immer überprüft werden, ob die URL in der Adresszeile des Browser auch tatsächlich der URL des echten OP entspricht.

Präventive Anmeldung beim OP: Der Benutzer navigiert zu Beginn seiner Onlineaktivitäten zu seinem OP und meldet sich dort an. Da die meisten OPs diese Anmeldung für einen längeren Zeitraum aufrecht erhalten können, kann der Benutzer davon ausgehen, dass er während seiner Sitzung von seinem OP nicht mehr nach den Zugangsdaten gefragt wird.

Browserplugins: Tools, wie das OpenID SeatBelt Plugin von VeriSign [Ver], die den Benutzer bei der Authentisierung gegenüber seinem OP unterstützten, können helfen unerwünschte Weiterleitungen zu verhindern.

OPs können ihre Benutzer vor Phishing Angriffen schützen, indem sie alternative und gegen Phishing resistent Authentifizierungsmechansimen in ihr System integrieren, wie zum Beispiel:

Digitale Zertifikate: Nach der Installation eines X.509 Zertifikats in den Browser des Benutzers, kann dieser über das installierte Zertifikat vom OP identifiziert werden ohne das die Eingabe eines Passworts erforderlich ist.

Challenge-Response Systeme: Wie in Abschnitt 2.2.2 erläutert kann mit dem Einsatz von Challenge-Response Systemen das Abhören von Zugangsdaten verhindert werden.

Anmeldung per Telefon: MyOpenID bietet seinen Benutzer die Möglichkeit sich per (Mobil-) Telefon anzumelden [Jana]. Auch hier entfällt die Notwendigkeit einer Passworteingabe.

6.2. Verfügbarkeit

Wie in allen SSO Umgebungen stellt auch der OP ein Single Point of Failure dar. Beim Ausfall des OP hat der Benutzer keine Möglichkeit sich mit seiner OpenID bei anderen RPs anzumelden. Noch fataler als ein temporärer Ausfall des OP ist dessen völlige Auflösung. Die Auflösung der Domain impliziert einen Verlust aller OpenIDs, die von diesem OP verwaltet wurden.

Abhilfe kann hier die Wahl eines etablierten OP oder die Delegation der eigenen OpenID an einen anderen OP schaffen. Mithilfe der Delegation kann der Benutzer eine URL als OpenID nutzen, die er selbst kontrolliert. Im Quellcode der Webseite, die über diese URL erreichbar ist, kann der Benutzer mit zwei `link` Tags im HTML-Kopf die URL seines OP angeben, an den alle Anfragen weitergeleitet werden sollen [Ope07, 7.3.3.]:

```
<link rel="openid2.provider" href="http://openid.provider.com" />
<link rel="openid2.local_id" href="http://openid.provider.com/userXY"/>
```

6.3. Benutzerfreundlichkeit

OpenID stellt ein völlig neues Konzept gegenüber den üblichen passwortbasierten Anmeldemechanismen dar, das teilweise ein Umdenken beziehungsweise Neulernen bei den Benutzern erfordert. Eine Usability Studie von Yahoo! [Yah] zeigt, dass viele Benutzer die Funktionsweise von OpenID auf den ersten Blick nicht klar durchschauen und als unnötig komplex empfinden. Trotz einheitlichem Design und der Einbindung der OpenID Logos, die für viele Benutzer noch unbekannt sind, fällt es vielen Benutzern schwer ein OpenID Eingabefeld mit ihrem OP in Verbindung zu setzten. Des Weiteren werden die vielen Einzelschritte und Weiterleitungen zwischen RP und OP, die bei einer OpenID basierten Anmeldung ablaufen als umständlicher Mehraufwand empfunden.

Trotzdem würdigen die Teilnehmer der Studie auch den Vorteil des webbasierten SSO Verfahren. Die Betreiber der Studie haben Empfehlungen ausgesprochen, wie man diesen negativen Eindrücken entgegen wirken kann.

Aufklärung und Verbreitung: Eine wirkliche Akzeptanz von OpenID ist nur möglich, wenn die Benutzer mit der Technologie vertraut sind und diese auch möglichst überall eingesetzt werden kann.

Wiedererkennung des OP: Für viele Benutzer ist es einfacher ein Anmeldeformular mit ihrem OP in Verbindung zusetzten, wenn dort das Logo ihres OP abgebildet ist. Formulare, wie sie zum Beispiel der frei verfügbare OpenID Selector [Opea] bereitstellt, können helfen die den Wiedererkennungswert zu steigern (siehe Abbildung 6.1).

Klare Abgrenzung: Das Eingabefeld für die OpenID sollte klar vom Formular zur passwortbasierten Anmeldung abgegrenzt werden, damit die Benutzer nicht verwirrt werden.

Möglichst geringer Aufwand: Die Interaktion mit dem Benutzer während einer OpenID basierten Authentifizierung sollte minimal gehalten werden. Die Möglichkeit bestimmte RPs

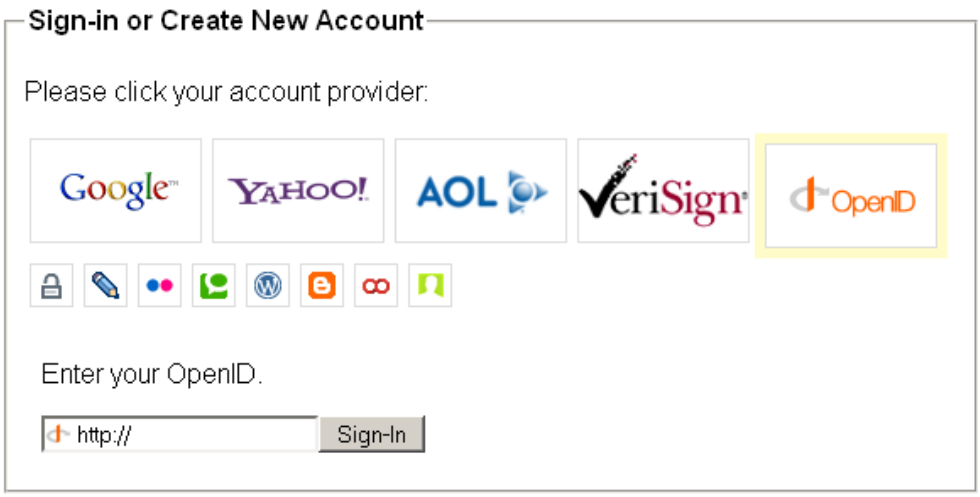


Abbildung 6.1.: Anmeldeformular mit Logos der OPs

als vertrauenswürdig einzustufen, sodass Authentifizierungsanfragen dieser RPs vom OP automatisch beantwortet werden können, steigern den Benutzerkomfort.

Teil III.

Integration von OpenID in Tricia

7. Übersicht

Aufbauend auf die theoretischen Grundlagen, die in den letzten beiden Teilen vermittelt wurden, wird in diesem Kapitel die Integration von OpenID in das Java-basierte Web Collaboration Tool *Tricia* dokumentiert.

Nach einer Beschreibung von Tricia in Abschnitt 8 folgt die Dokumentation der Integration von OpenID in Tricia, die angelehnt an [BD04] in vier Phasen durchgeführt wird: Anforderungsermittlung, Analyse, Architekturentwurf und Implementierung.

In Abschnitt 9 werden die funktionalen und nicht-funktionalen Anforderungen ermittelt und die möglichen Anwendungsfälle beschrieben. Anhand einer Analyse dieser Anforderungen und Anwendungsfälle wird in Abschnitt 10 ein Modell für die Anwendungsdomäne entworfen, das wiederum die Grundlagen für den in Abschnitt 11 beschriebenen Architekturentwurf liefert. Die konkrete Implementierung dieses Entwurfs wird abschließend in Abschnitt 12 dokumentiert.

8. Tricia

Aufbau des Kapitels Tricia ist ein Java-basiertes Web Collaboration Tool, über das mehrere Benutzer Information und Dokumente untereinander austauschen können. Tricia wurde auf Basis von Toro entwickelt, einer Plattform zur Entwicklung von dynamischen Webanwendungen.

In diesem Kapitel wird Tricia und das zugrunde liegende Toro Framework analysiert und beschrieben. Im ersten Teil werden die Systemarchitektur und grundlegende Komponenten (Assets, Templates, Handler und Plugins) vorgestellt. Da die Integration der OpenID Funktionalitäten unmittelbaren Einfluss auf die Benutzerverwaltung des Systems hat, werden im zweiten Teil relevante Aspekte der Benutzerverwaltung aufgeführt.

8.1. Architektur

Die Toro Plattform wurde nach dem Model-View-Controller (MVC) Architekturmuster [BD04, S. 239f] in drei Kerneinheiten strukturiert (siehe Abbildung 8.1). Das Datenmodell (engl. model) wird über *Assets*, die Präsentation (engl. view) über das *Toro Templating System* und die Programmsteuerung (engl. controller) durch die *Handler* realisiert.

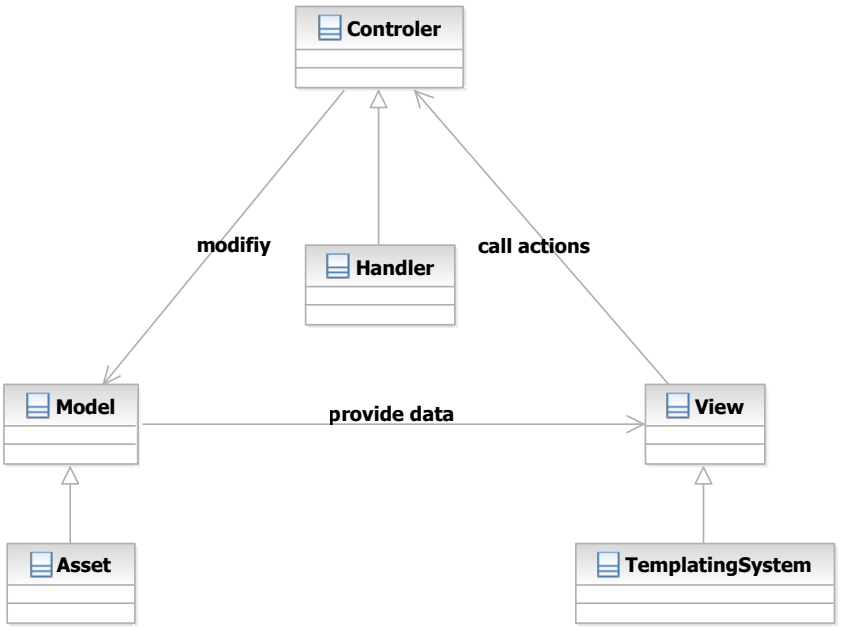


Abbildung 8.1.: Toro Architektur nach dem MVC Muster.

8.1.1. Datenmodell: Assets

Mit den Assets stellt die Toro Plattform einen objektrelationales Mapping zur Verfügung, mit dem Datenobjekte in einer relationaler Datenbank abgelegt werden können. Assets können Eigenschaften (*Properties*) bestimmter Typen (siehe Abbildung 8.2) und Beziehungen (*Roles*) untereinander haben (siehe Abbildung 8.3). Jeder Asset Klasse ist eine Tabelle in der

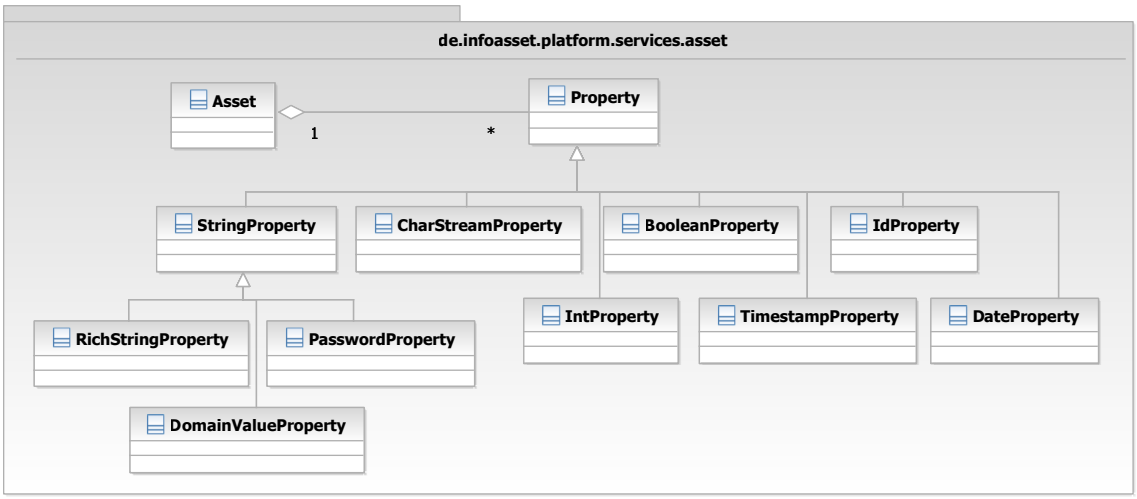


Abbildung 8.2.: Mögliche Eigenschaften eines Assets.

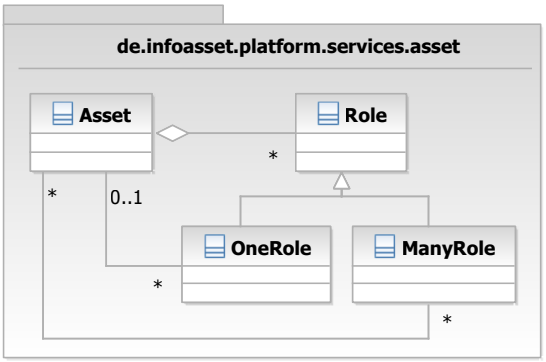


Abbildung 8.3.: Mögliche Beziehungen zwischen Assets.

relationalen Datenbank zugeordnet. Die Eigenschaften eines Assets entsprechen den Spalten der zugehörigen Tabelle. Beziehungen werden je nach Typ verschieden in der Datenbank abgebildet:

- *one-to-one* oder *one-to-many* Beziehungen werden durch einen Fremdschlüssel abgebildet, der auf den Beziehungspartner verweist.
- *many-to-many* Beziehungen werden in separaten Kreuztabellen abgebildet, in denen pro Eintrag zwei Fremdschlüssel existieren, die auf die jeweiligen Beziehungspartner

verweisen.

MixinAssets

Mit der *MixinAsset* Klasse, eine Unterklasse der Asset Klasse, lassen sich bestimmte Eigenschaften und Funktionalitäten einmalig implementieren, die dann für andere Assets wiederverwendet werden können. Mit der *CoreAsset.adapt(mixinClass)* Methode kann ein *CoreAsset* um die Eigenschaften und Funktionalitäten eines MixinAssets vom Typ *mixinClass* erweitert werden. Im Vergleich zur normalen Java-Vererbung können auf ein CoreAsset beliebig viele verschiedene MixinAssets angewendet werden, ohne das diese mit einander in Beziehung stehen müssen. In Abbildung 8.4 ist der Zusammenhang zwischen Assets, CoreAssets und MixinAssets nocheinmal vereinfacht dargestellt.

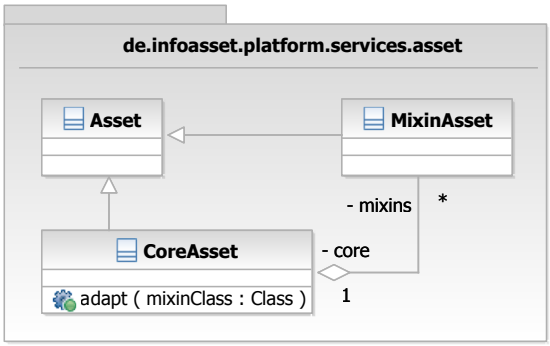


Abbildung 8.4.: Beziehung zwischen CoreAssets und MixinAssets

8.1.2. Präsentation: Templating System

Mit dem Toro Templating System können dynamische Webseiten generiert werden, die in Webapplikationen einer grafischen Benutzeroberfläche entsprechen. Zwischen Darstellung und Logik findet eine strikte Trennung statt: In den Templates, die in der Regel HTML Code enthalten, sind bestimmte Platzhalter definiert, die vom Templating System dynamisch mit den passenden Daten ersetzt werden. Für diese Platzhalter existieren vier grundlegende Typen:

- *Print Substitutions* sind die einfachsten Platzhalter, die lediglich durch eine Zeichenkette ersetzt werden.
- *Conditional Substitutions* bieten die Möglichkeit die Templateausgabe nach bestimmten Bedingungen zu gestalten.
- *List Substitutions* dienen dazu einen Block mit Datenelementen beliebiger Anzahl auszugeben, die jeweils in gleicher Form dargestellt werden sollen.
- *Template Substitutions* ermöglichen das Einbinden von anderen Templates in einem Template.

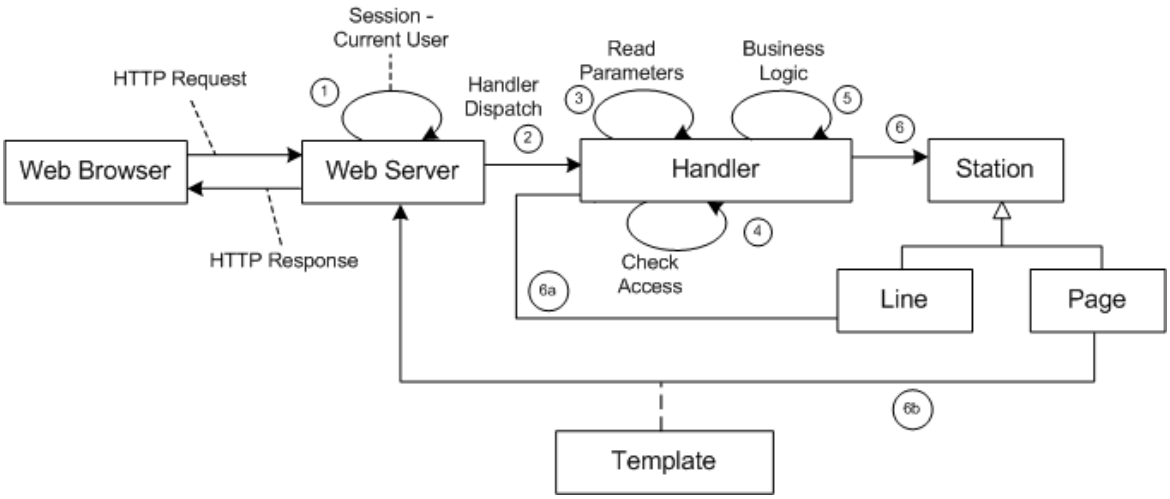


Abbildung 8.5.: Schematischer Ablauf der Verarbeitung einer Anfrage.

8.1.3. Programmsteuerung: Handler

Die Handler werten HTTP Anfragen aus und generieren eine dazu passende Antwort. Der Ablauf einer solchen Anfrage ist in Abbildung grafisch dargestellt und beinhaltet folgende Schritte:

1. Der *HttpServer* (Webserver) identifiziert die Session und erzeugt einen Client-Kontext.

2. Der *HandlerDispatcher* ermittelt den *Handler*, der für die Bearbeitung der Anfrage zuständig ist.

3. Der Handler ermittelt in der Methode *getParameters(ParameterReader)* die in der Anfrage enthaltenen und für die Verarbeitung nötigen Parameter.

4. Mit der Methode *checkAccess()* prüft der Handler, ob der Besitzer der Session Zugriff auf den Handler hat.

5. Der Handler führt die eigentliche Applikationslogik aus, indem die Methode *doBusinessLogic()* ausgeführt wird. Diese Methode ist *abstract* und muss von jeder Handler Klasse implementiert werden. Sie liefert als Rückgabewert ein Objekt vom Typ *Station* zurück.

6. Die Station leitet die Anfrage entweder zu einen anderen Handler weiter (6a), oder liefert eine Antwort in Form einer Seite, die über ein Template erzeugt wird (6b).

HandlerDispatcher

Bei einer HTTP Anfrage ermittelt der *HandlerDispatcher* den Handler, der für die Verarbeitung der Anfrage zuständig ist. Dies wird durch ein URL Mapping realisiert, das nach zwei Methoden funktioniert:

Mapping von statischen URLs über festgelegte Konventionen: Hier stimmt der Name der Handler Klasse direkt mit einem Teil des Pfads überein. Für die Ermittlung des Handlers wird der Pfad im ersten Schritt aufgearbeitet und alle Slashes (/) durch Punkte (.) ersetzt. Vom letzten Teil des Pfads wird der Anfangsbuchstabe in ein Großbuchstaben umgewandelt und der Suffix *Handler* angehängt. Die so erhaltene Zeichenkette entspricht dann dem Klassennamen des Handlers im Package `de.infoasset.PLUGINNAME.handler`.

So wird z.B. aus dem Pfad *document/edit* der Klassenname `de.infoasset.PLUGINNAME.handler.document.EditHandler` ermittelt. Im letzten Schritt versucht der Dispatcher in den passenden Plugins die Handler Klasse zu finden und bindet diese dynamisch zur Laufzeit ein.

Mapping von dynamischen URLs über reguläre Ausdrücke: Bei diesem Verfahren kommen URLs zum Einsatz die für den Benutzer lesbarer als die statischen und technischen URLs sind. Mit *HandlerPattern* Klassen können diese URLs dann auf den jeweiligen Handler abgebildet werden.

8.1.4. Plugins

Die Toro Plattform ist modular aufgebaut und kann durch *Plugins* erweitert werden. Kern der Plattform bildet das *toro* Plugin, das die grundlegenden Funktionalitäten der Plattform bereitstellt, wie zum Beispiel die Benutzerverwaltung.

Für die Entwicklung von Plugins existieren Konventionen, die die Struktur eines Plugins festlegen:

- Konfigurations Dateien werden im Pluginverzeichnis unter *config/* abgelegt.
- Templates befinden sich im Pluginverzeichnis unter *templates/*.
- Externe Bibliotheken werden im Pluginverzeichnis unter *lib/* abgelegt.
- Der Java Sourcecode befindet sich im Pluginverzeichnis unter *src/classes/*.
- Alle Klassen des Plugins befinden sich im Package `de.infoasset.PLUGINNAME`.
- Die Asset Klassen des Plugins befinden sich im Package `de.infoasset.PLUGINNAME.assets`.
- Die Handler Klassen des Plugins befinden sich im Package `de.infoasset.PLUGINNAME.handler`.

8.2. Benutzerverwaltung

In Abbildung 8.6 sind die Assets des Toro Plugins abgebildet, die für die Benutzerverwaltung von besonderer Bedeutung sind.

Zentrale Datenobjekte der Benutzerverwaltung sind die *Person* und *Group* Assets, die Benutzer beziehungsweise Benutzergruppen repräsentieren. Über die Benutzergruppen können mehrere Benutzer zusammengefasst organisiert werden. Grundlage für das Person und Group Asset stellt das *Principal* Asset dar, das unter anderen folgende Eigenschaften bereitstellt:

- *name*: Der eindeutiger Name des Objekts.

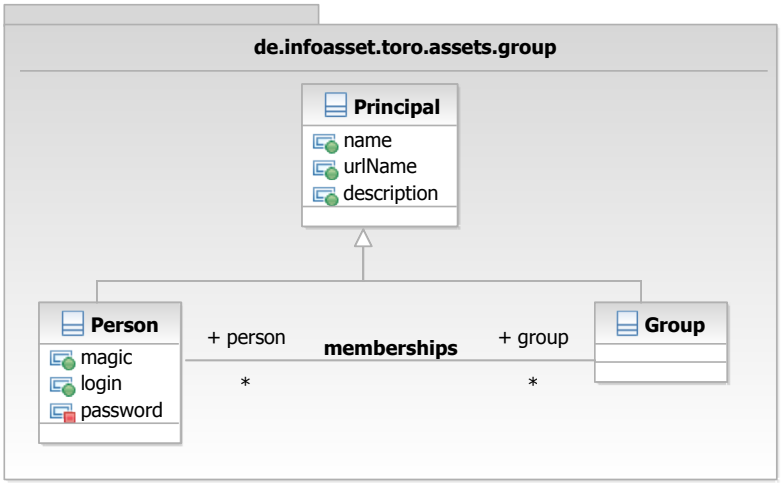


Abbildung 8.6.: Konzept der Benutzerverwaltung des Toro Plugins

- *urlName*: Formatierter Name, so dass dieser in eine URL eingebettet werden kann. Diese Eigenschaft wird später Grundlage für die Generierung der OpenIDs für die lokalen Benutzerkonten sein (siehe Kapitel 11).
- *description*: Eine Beschreibung des Objekts.

Das Person Asset stellt weitere Eigenschaften bereit, die für die Integration der OpenID Funktionalitäten von Bedeutung sind:

- *magic*: Eine zufällig generierte Zeichenkette, die gesetzt wird wenn das Benutzerkonto vom Benutzer verifiziert werden muss. Dem Benutzer wird dazu ein Link, der die generierte Zeichenkette enthält per E-Mail gesendet. Nach dem Aufruf des Links durch den Benutzer, wird die übermittelte Zeichenkette mit der gespeicherten verglichen. Stimmen die beiden Zeichenketten überein, ist das Konto verifiziert und die Zeichenkette wird wieder entfernt.
- *login*: Die eindeutige Anmeldekennung des Benutzers, die in Form einer E-Mail vorliegt.
- *password*: Das MD4 [Gro] verschlüsseltes Passwort des Benutzers.

8.2.1. Registrierung

Ein Benutzer ohne Konto hat die Möglichkeit über den Link im oberen Bereich der Webseite (siehe Abbildung 8.7) die Registrierung einzuleiten.

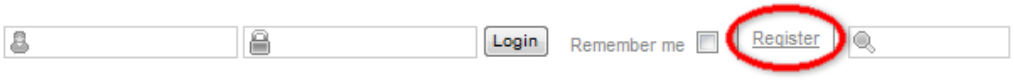


Abbildung 8.7.: Link zur Registrierung im oberen Bereich der Webseite

Die Registrierung verläuft in vier Schritten ab:

1. **Datenerfassung**: Das System stellt dem Benutzer ein Formular zur Eingabe seiner persönlichen Daten (Name, E-Mail und Passwort) bereit.

2. **Datenprüfung**: Das System überprüft die Gültigkeit der eingebenden Daten. Sind die Daten fehlerhaft, wird der Benutzer wieder zu Schritt 1 weitergeleitet.

3. **Erstellung des Kontos**: Das System legt ein Benutzerkonto mit den erfassten Daten an, indem ein neues Person Assets erstellt wird. Beim Erstellen des Assets wird automatisch die *magic* Eigenschaft mit einer zufälligen Zeichenkette belegt. Auf Basis dieser Zeichenkette wird eine URL zur Aktivierung des Benutzerkontos generiert, die an die E-Mail Adresse des Benutzers gesendet wird.

4. **Aktivierung des Kontos**: Der Benutzer aktiviert sein Konto durch den Aufruf der in der E-Mail enthaltenen URL.

8.2.2. Anmeldung

Ein registrierter und unangemeldeter Benutzer hat die Möglichkeit sich über den passwortbasierten Anmeldemechanismus beim System anzumelden. Dazu stehen im oberen Bereich der Webseite (siehe Abbildung 8.8) zwei Felder zur Eingabe der Anmeldekennung und des Passworts zur Verfügung. Wie bereits erwähnt dient die E-Mail Adresse eines Benutzers als Anmeldekennung. Nach Betätigung des *Login* Buttons durch den Benutzer und erfolgreicher

Abbildung 8.8.: Eingabefelder zur Anmeldung im oberen Bereich der Webseite

Verifizierung der Eingabedaten durch das System ist der Benutzer angemeldet.

9. Anforderungen

Aufbau des Kapitels In diesem Kapitel werden die Anforderungen für die Integration von OpenID in das Tricia System näher untersucht. In den ersten beiden Abschnitten werden die funktionalen und nicht-funktionalen Anforderungen beschrieben, die Grundlage für die Anwendungsfälle in Abschnitt 9.3 sind.

9.1. Funktionale Anforderungen

Das System soll die OpenID Funktionalität komplett unterstützen, das heißt sowohl als RP als auch als OP fungieren können. Daraus ergeben sich folgende konkreten funktionale Anforderungen:

In der Rolle als Relying Party

- Ein Benutzer soll seine OpenIDs von fremden OPs mit seinem lokalen Konto verknüpfen können.
- Ein Benutzer soll sich mit einer verknüpften OpenID beim System anmelden können.
- Ein Benutzer soll sich mit einer OpenID eines fremden OP beim System registrieren können ohne seine Benutzerdaten dabei angeben zu müssen.

In der Rolle als OpenID Provider

- Jedes lokale Benutzerkonto soll mit einer vom System verwalteten OpenID ausgestattet sein.
- Ein Benutzer soll Anfragen von fremden RP auf seine lokale OpenID zu einem fremden OP weiterleiten können.
- Ein Benutzer soll verschiedene OpenID Profile für seine lokale OpenID verwalten können. Mit diesen Profilen soll der Benutzer bei einer Anfrage durch eine fremde RP entscheiden können, welche Attribute an die anfragende RP weitergeleitet werden sollen.
- Ein Benutzer soll sich mit seiner lokalen OpenID bei fremden RPs authentifizieren und gegebenenfalls Profildaten an diese RPs senden können.
- Ein Benutzer soll in der Lage sein fremde RPs als vertrauenswürdig einstufen zu können, sodass Anfragen dieser RPs vom System automatisch und ohne explizite Bestätigung durch den Benutzer beantwortet werden können.

9.2. Nicht-Funktionale Anforderungen

Neben den funktionalen Anforderungen sollen folgende Punkte bei der Integration der OpenID Funktionalität in das System beachtet werden:

Benutzerfreundlichkeit

Für den Benutzer soll schnell und klar erkennbar sein wo sich OpenID Funktionalitäten befinden. Um das zu erreichen, sollten OpenIDs, Links, Formulare und Eingabefelder immer mit dem in Abbildung 9.1 dargestellten offiziellen OpenID Logo gekennzeichnet werden.

Abbildung 9.1.: Offizielles OpenID Logo

Damit der Browser und eventuell installierte Browsererweiterungen ein OpenID Eingabefeld automatisch identifizieren können, sollte der Name dieser Eingabefelder stets mit dem in der OpenID Spezifikation [Ope07][7.1.] empfohlenen Wert „*openid_identifier*“ belegt werden.

Implementierung

Da Tricia in Java geschrieben ist, muss auch bei der Implementierung der OpenID Unterstützung Java verwendet werden.

Die Integration von OpenID soll über eines eigenständigen Plugins realisiert werden, sodass Tricia wahlweise mit oder ohne OpenID Unterstützung betrieben werden kann. Bei der Entwicklung des Plugins sollen die in Abschnitt 8.1.4 aufgeführten Konventionen eingehalten und, sofern anwendbar, schon existierende Funktionalitäten und Schnittstellen der Toro Plattform genutzt werden.

OpenID Bibliothek

Zur Bereitstellung grundlegender Funktionalitäten des OpenID Protokolls soll eine bereits verfügbare Bibliothek verwendet werden. Auch hier wird gefordert, dass diese Bibliothek in Java geschrieben ist. Die Einbindung der Bibliothek sollte ohne größere Anpassungen an der Bibliothek möglich und in einer ausführlichen und gut strukturierten Dokumentation beschrieben sein.

Nach Möglichkeit sollten die OpenID Spezifikationen vollständig implementiert sein. Zwingend erforderlich ist die Implementierung der OpenID Authentication 2.0 Spezifikation [Ope07] sowie der OpenID Simple Registration Extension 1.0 Protokollerweiterung [HDJ+06].

9.3. Anwendungsfälle

Anhand der funktionalen Anforderungen aus Abschnitt 9.1 lassen sich die in Abbildung 9.2 zusammengefassten Anwendungsfälle herleiten. Als mögliche Akteure treten die Benutzer

des Tricia Systems, OPs, die in einem Vertrauensverhältnis zu den Tricia Systembetreibern stehen, und RPs auf, die das Tricia System als OP nutzen.

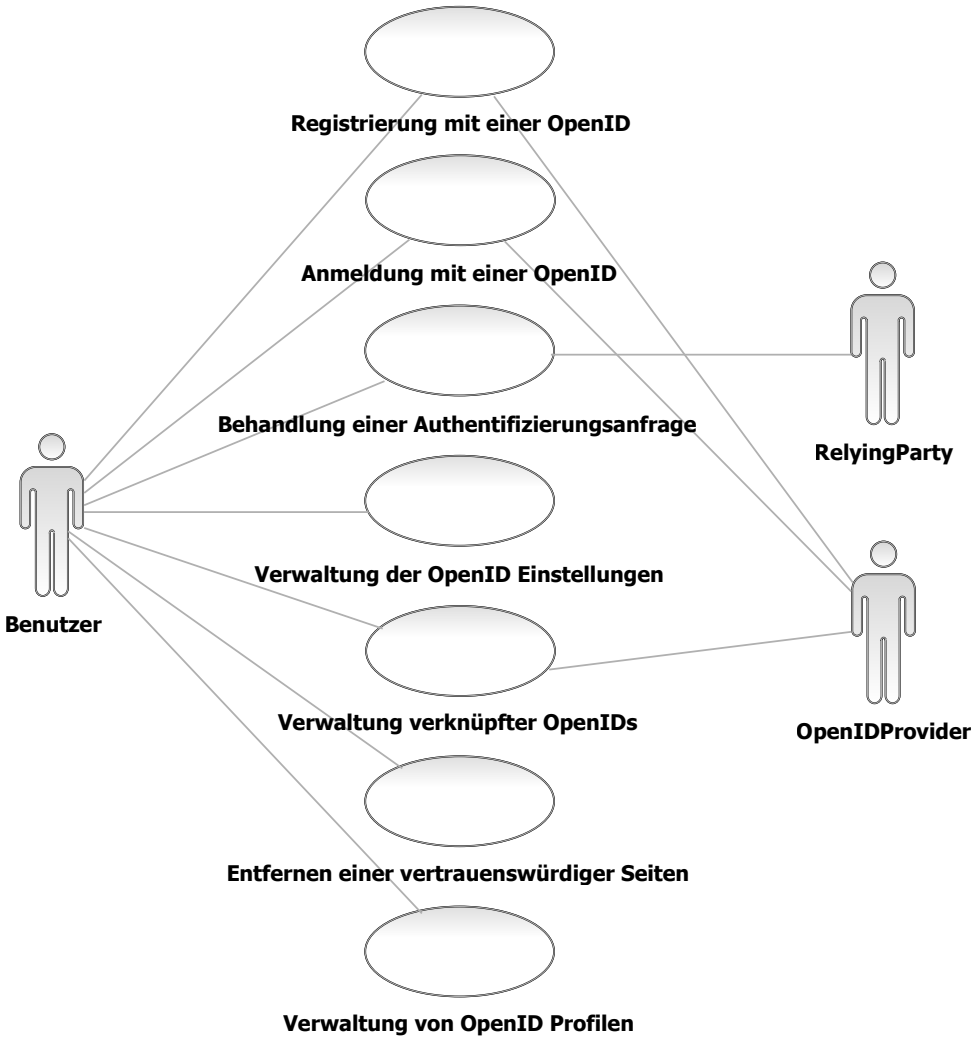


Abbildung 9.2.: Anwendungsfälle für die OpenID Funktionalität in Tricia

Nachfolgend werden alle Anwendungsfälle kurz beschrieben. Detaillierte Informationen zu den einzelnen Anwendungsfällen können dem Abschnitt A im Anhang entnommen werden.

A.1: Registrierung mit einer OpenID

Ein Benutzer registriert ein neues lokales Benutzerkonto durch die Angabe einer OpenID.

A.2: Anmeldung mit einer OpenID

Ein Benutzer meldet sich mit einer OpenID eines anderen OP beim System an.

Behandlung einer Authentifizierungsanfrage

Das System muss drei verschiedenen Authentifizuierungsanfragen behandeln können, die in Abbildung 9.3 schematisch dargestellt sind:

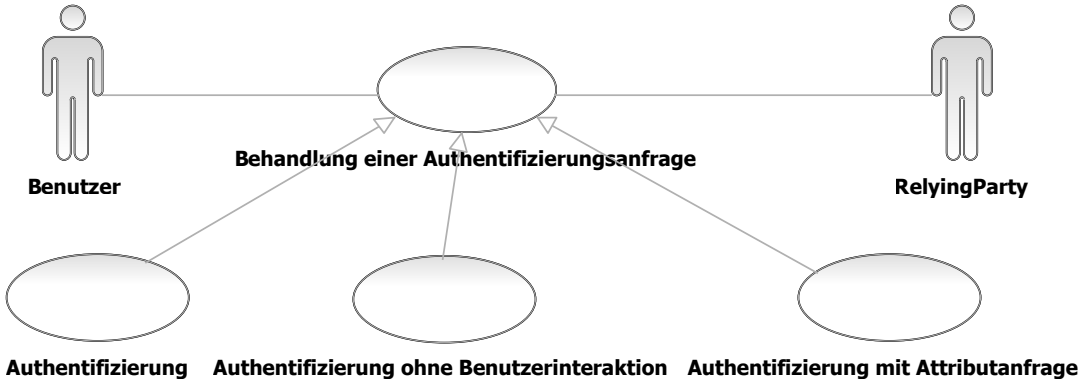


Abbildung 9.3.: Anwendungsfälle zur Behandlung von Authentifizierungsanfragen

A.3: Authentifizierung

Die RP fordert eine Bestätigung an, dass der Benutzer Inhaber der angegebene OpenID ist.

A.4: Authentifizierung mit Attributanfrage

Die RP fragt nach einer Bestätigung, dass der Benutzer Inhaber der angegebene OpenID ist und fordert zusätzlich bestimmte Attribute des Benutzer an.

A.5: Authentifizierung ohne Benutzerinteraktion

Die RP stellt eine Authentifizierungsanfrage mit oder ohne Attributanfrage, die vom System durch zuvor gespeicherten Einstellungen automatisch und ohne Interaktion mit dem Benutzer beantwortet wird.

A.6: Verwaltung der OpenID Einstellungen

Ein Benutzer konfiguriert seine persönlichen OpenID-Einstellungen:

- Weiterleitung der lokalen OpenID zu einer OpenID bei einem anderen OP.
- Aktivierung/Deaktivierung lokalen passwortbasierten Anmeldemechanismus

Verwaltung verknüpfter OpenIDs

Der Anwendungsfall zur Verwaltung verknüpfter OpenIDs lässt sich wie in Abbildung 9.4 dargestellt in zwei separate Anwendungsfälle unterteilen:

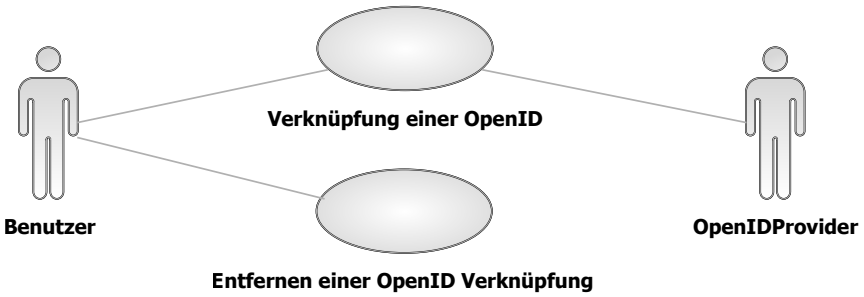


Abbildung 9.4.: Unterteilung des Anwendungsfalls zur Verwaltung verknüpfter OpenIDs

A.7: Verknüpfung einer OpenID

Ein Benutzer verknüpft eine OpenID mit seinem lokalen Benutzerkonto.

A.8: Entfernen einer OpenID Verknüpfung

Ein Benutzer entfernt eine Verknüpfung zwischen seinem lokalen Benutzerkonto und einer OpenID.

A.9: Entfernen einer vertrauenswürdigen Seite

Ein Benutzer entfernt eine Seite, die er in der Vergangenheit als vertrauenswürdig eingestuft hat.

Verwalten von OpenID Profilen

Die Verwaltung von OpenID Profilen umfasst drei Anwendungsfälle, die in Abbildung 9.5 schematisch dargestellt sind:

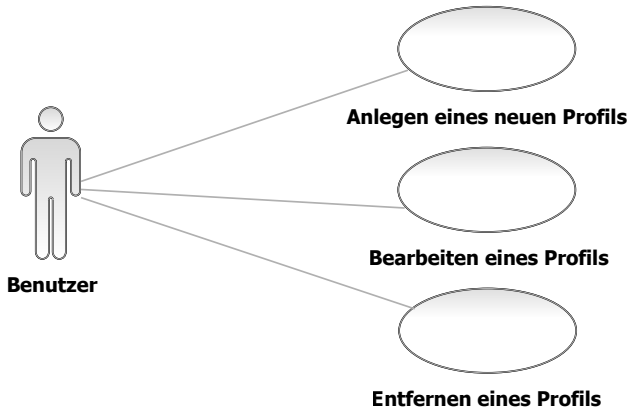


Abbildung 9.5.: Anwendungsfälle zur Verwaltung von OpenID Profilen

A.10: Anlegen eines neuen Profils

Ein Benutzer erstellt ein neues OpenID Profil.

A.11: Bearbeiten eines Profils

Ein Benutzer bearbeitet eines seiner OpenID Profile.

A.12: Entfernen eines Profils

Ein Benutzer entfernt eines seiner OpenID Profile.

10. Analyse

Aufbau des Kapitels In diesem Kapitel werden die Anforderungen und Anwendungsfälle aus dem vorherigen Kapitel analysiert um ein Modell zu entwickeln, das die Objekte der Anwendungsdomäne und deren Beziehungen unter einander näher beschreibt.

10.1. Benutzer, Benutzerkonten und OpenIDs

Zentrale Objekte stellen Benutzer, Benutzerkonten und OpenIDs dar. Ein Benutzer entspricht einer realen Person, die durch Registrierung beim System in den Besitz eines Benutzerkontos kommen kann. Wichtige Attribute des Benutzerkontos sind der Name, sowie die Email-Adresse und das Passwort. Mithilfe der Email-Adresse und des Passworts kann ein registrierter Benutzer sich über den lokalen Anmeldemechanismus beim System anmelden. Wichtigstes Attribut einer OpenID ist der Identifier, mit dem die OpenID eindeutig identifiziert werden kann. In der Regel liegt der Identifier in Form einer URL vor (siehe dazu Kapitel 4).

In Abbildung 10.1 sind die Beziehungen zwischen Benutzern, Benutzerkonten und OpenIDs grafisch dargestellt: Ein registrierter Benutzer besitzt genau ein Benutzerkonto. Diesem

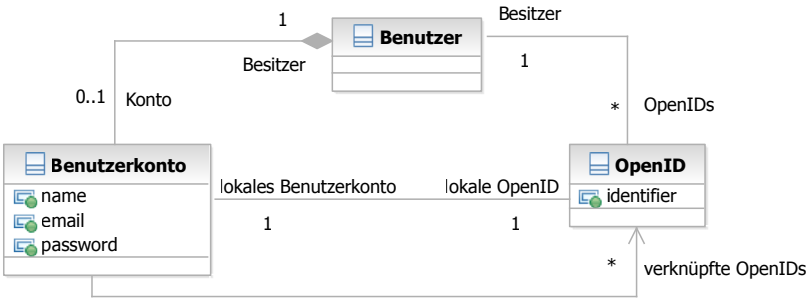


Abbildung 10.1.: Beziehungen zwischen Benutzern, Benutzerkontonen und OpenIDs

Benutzerkonto ist eine lokale OpenID zugeordnet, die vom System verwaltet wird. Zusätzlich kann ein Benutzer über weitere OpenIDs verfügen, die von fremden OPs verwaltet werden. Ein registrierter Benutzer kann OpenIDs, die von fremden OPs verwaltet werden, mit seinem lokalen Benutzerkonto verknüpfen, sodass er sich mit diesen OpenIDs beim System anmelden kann.

10.2. OpenID Profile

Nutzt ein Benutzer das System als OP, kann er seine Benutzerdaten zentral über sein Benutzerkonto pflegen. Da ein Benutzer bei anderen RP in verschiedenen Rollen auftreten kann, z.B. als Privatperson oder Geschäftskunde, oder nur eine bestimmte Auswahl seiner

Benutzerdaten an Dritte weiterreichen möchte, kann er auf dem System verschiedene Profile verwalten. In jedem Profil kann der Benutzer dann Angaben zu seiner Person hinterlegen (siehe Abbildung 10.2). Fordert eine RP eine Authentifizierung mit Attributanfrage an (siehe

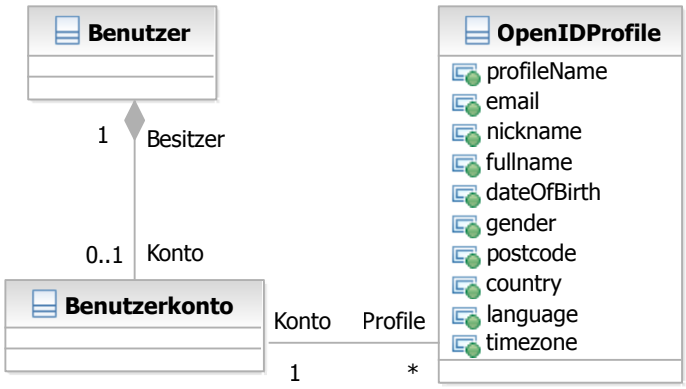


Abbildung 10.2.: Beziehungen zwischen Benutzer, Benutzerkonto und Profilen

Tabelle A.4), kann der Benutzer über die Auswahl eines Profils entscheiden, welche Daten vom System an die RP übertragen werden sollen. In einem Profil können die Attribute belegt werden, die mit Hilfe der OpenID Simple Registration Extension Protokollerweiterung übertragbar sind (siehe Abschnitt 5.2.1).

10.3. Vertrauenswürdige Seiten

Authentifiziert das System den Benutzer für eine fremde RP kann dieser die Anfrage der RP ablehnen, einmalig oder dauerhaft genehmigen (vergleiche Anwendungsfall „Authentifizierung“). Genehmigt der Benutzer die Anfrage dauerhaft speichert das System die anfragende RP als vertrauenswürdige Seite ab. Stellt eine vertrauenswürdige Seite eine Authentifizierungsanfrage an das System, wird diese Anfrage automatisch ohne eine Bestätigung des Benutzers positiv beantwortet.

Genehmigt der Benutzer eine Authentifizierung mit Attributanfrage dauerhaft (siehe Tabelle A.4), speichert das System zusätzlich auch das vom Benutzer gewählte Profil, dessen Daten an die anfragende RP gesendet wird. Eine Authentifizierung mit Attributanfrage für eine vertrauenswürdige Seite wird vom System nur dann automatisch beantwortet, wenn der Benutzer bei der dauerhaften Genehmigung ein Profil ausgewählt hat.

In Abbildung 10.3 sind die Zusammenhänge zwischen Benutzer, Benutzerkonto, vertrauenswürdigen Seiten und OpenID Profilen grafisch dargestellt: Die vertrauenswürdigen Seiten eines Benutzers sind dessen Benutzerkonto zugeordnet. Für jede vertrauenswürdige Seite wird der Name sowie der Zeitpunkt der letzten Authentifizierung gespeichert. Zusätzlich wird ein Verweis zu dem OpenID Profil gespeichert, dass der vertrauenswürdigen Seite bei der dauerhaften Genehmigung zur Verfügung gestellt wurde.

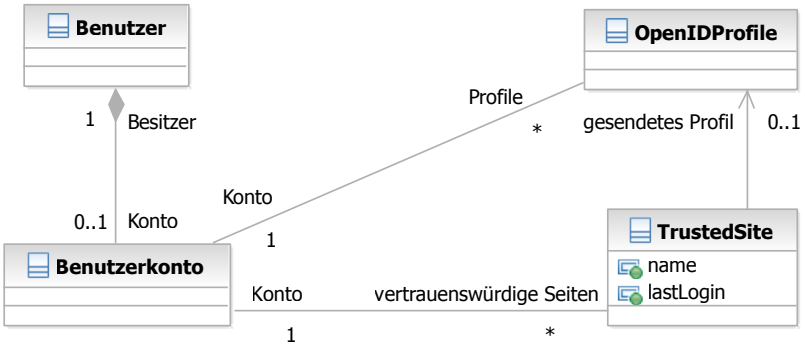


Abbildung 10.3.: Beziehungen zwischen Benutzer, Benutzerkonto, vertrauenswürdigen Seiten und Profilen

11. Architekturentwurf

Aufbau des Kapitels In diesem Kapitel wird die grundlegende Planung der OpenID Integration in Tricia beschrieben. Im ersten Teil wird die Auswahl einer geeigneten OpenID Bibliothek durchgeführt, mit der die OpenID Funktionalitäten implementiert werden sollen. Der Entwurf des OpenID Plugins wird im zweiten Teil dieses Kapitels behandelt.

11.1. Auswahl einer geeigneten OpenID Bibliothek

Die OpenID Foundation stellt unter [Fou] eine Übersicht derzeit verfügbarer Implementierungen in gängigen Programmiersprachen bereit. Nach der Vorauswahl anhand der in Abschnitt 9.2 formulierten Anforderungen an die Bibliothek stehen noch folgende Java-Implementierungen zur nähren Auswahl, die alle die Funktionalitäten für den Betrieb einer RP und eines OP bereitstellen:

- *JOID* [joi]
- *OpenID4Java* [opeb]
- *WSO2 Identity Server* [WSO]
- *NetMesh InfoGrid LID* [Net]

Allerdings sprechen einige Punkte gegen den Einsatz des WSO2 Identity Servers oder des NetMesh InfoGrids. So ist der WSO2 Identity Server ein Open-Source Identity Management Server. Eine spezielle Dokumentation der OpenID-Komponente ist nicht vorhanden, was vermuten lässt, dass der separate Einsatz dieser Komponente in anderen Systemen nicht vorgesehen ist. Ähnlich verhält es sich bei NetMesh InfoGrid LID, das ein Paket des NetMesh InfoGrid Frameworks zur Entwicklung von Webapplikationen ist. Auch hier existiert keine genauere Dokumentation, wie die OpenID-Komponente in andere Systeme eingebunden werden kann.

JOID und OpenID4Java sind hingegen eigenständige OpenID-Bibliotheken, die ausschließlich die OpenID Spezifikationen implementieren und für die Einbindung in andere Systeme entwickelt wurden. Allerdings ist OpenID4Java gegenüber JOID vorzuziehen. So implementiert OpenID4Java die OpenID Spezifikationen vollständig, wohingegen JOID derzeit noch keine Unterstützung für das Auflösen von XRI URLs sowie für die Übergabe von Klartext MAC keys bietet (z.B. bei der Verwendung von SSL verschlüsselten HTTP-Verbindungen). Die von JOID nicht unterstützten Features sind zwar nicht zwingend erforderlich, lassen jedoch einen höheren Reifegrad der OpenID4Java Bibliothek vermuten.

Auch die Dokumentation von JOID bietet im Vergleich zu OpenID4Java weniger Informationen für Entwickler. So ist zum Beispiel in der Dokumentation zu JOID nicht klar

beschrieben, wie die Simple Registration Extension (siehe Abschnitt 5.2.1) realisiert ist beziehungsweise wie diese mit der Bibliothek genutzt werden kann[1]. Da diese OpenID Protokollerweiterung zur Anforderung von Attributen im Anwendungsfall „Registrierung mit einer OpenID“ zum Einsatz kommt (siehe Tabelle A.1), ist eine gute Dokumentation an dieser Stelle von hohem Nutzen.

Das reichhaltigere und aktuellere Informationsangebot auf der Projektseite von OpenID4Java lässt zusätzlich auf eine größere und aktivere Community schließen, was der Weiterentwicklung und Verbesserung der Bibliothek auch in Zukunft zu Gute kommen kann.

11.1.1. OpenID4Java

Die OpenID4Java Bibliothek stellt zwei sogenannten *Manager* zur Verfügung (siehe Abbildung 11.1). Über diese Manager lassen sich OpenID Nachrichten erstellen, verarbeiten und versenden.

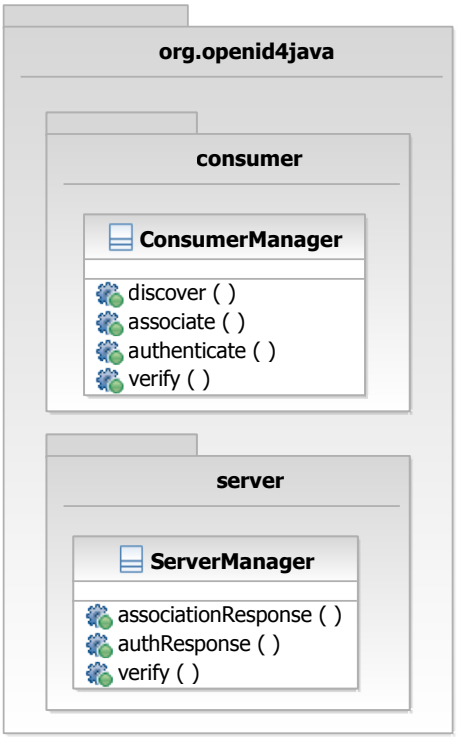


Abbildung 11.1.: ConsumerManager und ServerManager der OpenID4Java Bibliothek

Der *ConsumerManager* stellt unter anderen folgende Methoden zum Betrieb einer RP bereitstellt:

- *discover()*: Normalisiert einen OpenID Identifier und ermittelt den zugehörigen OP (entspricht Schritt 2 in Abbildung 5.1).

[1] Einziger Kommentar dazu im Quellcode der Klasse *SimpleRegistration*:„Simple registration extensions, as defined by http://openid.net/specs/openid-simple-registration-extension-1_0.html. This class should only be used by internal request/response processing. TODO to make this clearer.“

- *associate()*: Tauscht einen gemeinsamen Schlüssel mit einem OP aus (entspricht Schritt 3 in Abbildung 5.1).
- *authenticate()*: Generiert eine Authentifizierungsanfrage, die an einen OP gesendet werden kann (entspricht Schritt 4 in Abbildung 5.1).
- *verify()*: Verifiziert die Antwort eines OP auf eine Authentifizierungsanfrage (entspricht Schritt 7 in Abbildung 5.1).

Der *ServerManager* stellt unter anderen folgende Methoden zum Betrieb eines OP bereitstellt:

- *associationResponse()*: Generiert eine Antwort auf eine Anfrage zum Austausch eines gemeinsamen Schlüssels zwischen RP und OP (Schritt 3 in Abbildung 5.1).
- *authResponse()*: Generiert eine Antwort auf eine Authentifizierungsanfrage (entspricht Schritt 6 in Abbildung 5.1).
- *verify()*: Generiert eine Antwort auf eine Verifikationsanfrage einer RP (entspricht Schritt 7 in Abbildung 5.1).

11.2. Grundlegender Aufbau

Damit Tricia flexibel mit oder ohne OpenID Unterstützung gestartet werden kann, werden die OpenID Funktionalitäten in Form eines Plugins in das System integriert. Der grundlegende Aufbau des OpenID Plugins ist in Abbildung 11.2 grafisch dargestellt.

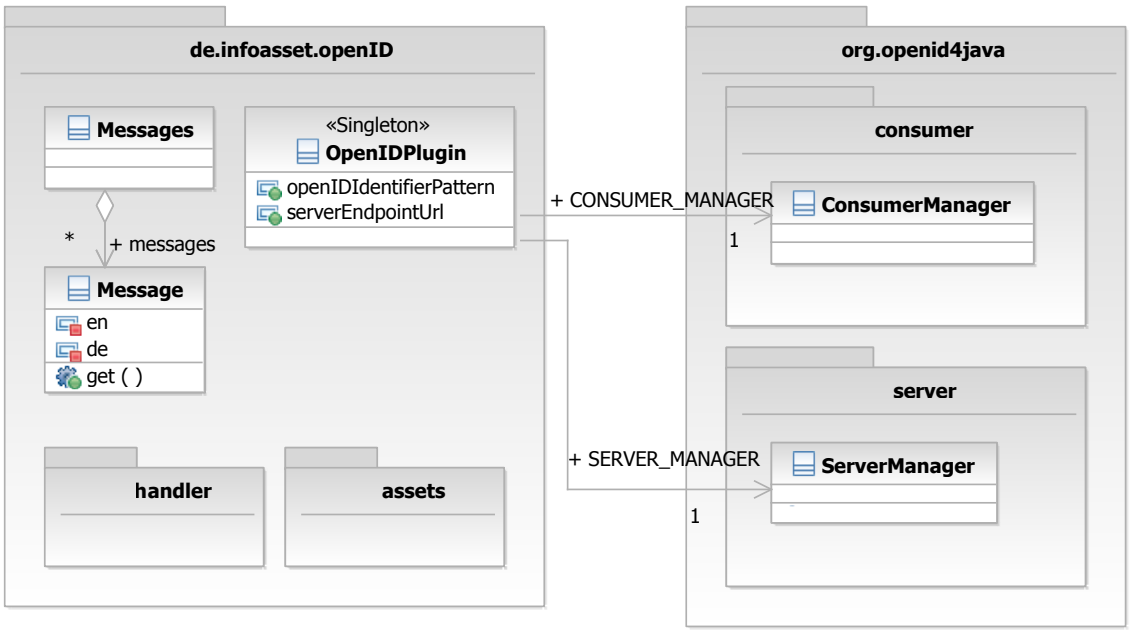


Abbildung 11.2.: Grundlegender Aufbau des OpenID Plugins

11.2.1. OpenID Plugin

Das Plugin wird über die Singleton-Klasse *OpenIDPlugin* verwaltet und gesteuert. Diese Klasse hat unter anderem folgende Attribute:

- *serverEndpointUrl*: Definiert die URL, über die fremde RPs OpenID Anfragen an das System senden können.
- *openIDIdentifierPattern*: Anhand dieses Musters werden die OpenIDs der lokalen Benutzerkonten generiert. In dem Muster muss ein Platzhalter {*urlName*} enthalten sein. Dieser Platzhalter wird bei der Generierung einer lokalen OpenID mit dem Namen des jeweiligen Benutzers ersetzt wird. Die komplette OpenID ergibt sich dann durch das Anhängen des Musters an den Hostnamen des System. Auf einem Host *http://tricia* mit einem *openIDIdentifierPattern* „/~{*urlName*}“ würde die OpenID des Benutzers „mustermann“ zum Beispiel *http://tricia/~mustermann* lauten.
- *CONSUMER_MANAGER*: Eine Instanz der *ConsumerManager* Klasse aus der OpenID4Java Bibliothek.
- *SERVER_MANAGER*: Eine Instanz der *ServerManager* Klasse aus der OpenID4Java Bibliothek.

11.2.2. Unterstützung von Mehrspachigkeit

In der Klasse *Messages* wird für jede Textausgabe ein mehrsprachiges Message Objekt definiert, das über die *get()* Methode den Inhalt der betreffenden Ausgabe in der aktuell gewählten Sprache zurück gibt.

12. Implementierung

Aufbau des Kapitels In diesem Kapitel wird die konkrete Umsetzung des Modells aus dem vorherigen Kapiteln beschrieben. Im ersten Teil werden die Assets beschrieben, die zur persistenten Datenspeicherung genutzt werden. Im zweiten Teil folgt eine Beschreibung der Handler, die für die Steuerung und und Ausführung der Applikationslogik verantwortlich sind.

12.1. Assets

Abbildung 12.1 zeigt den Aufbau des `de.infoasset.openid.assets` Packages, indem sich die Assets des OpenID Plugins befinden.

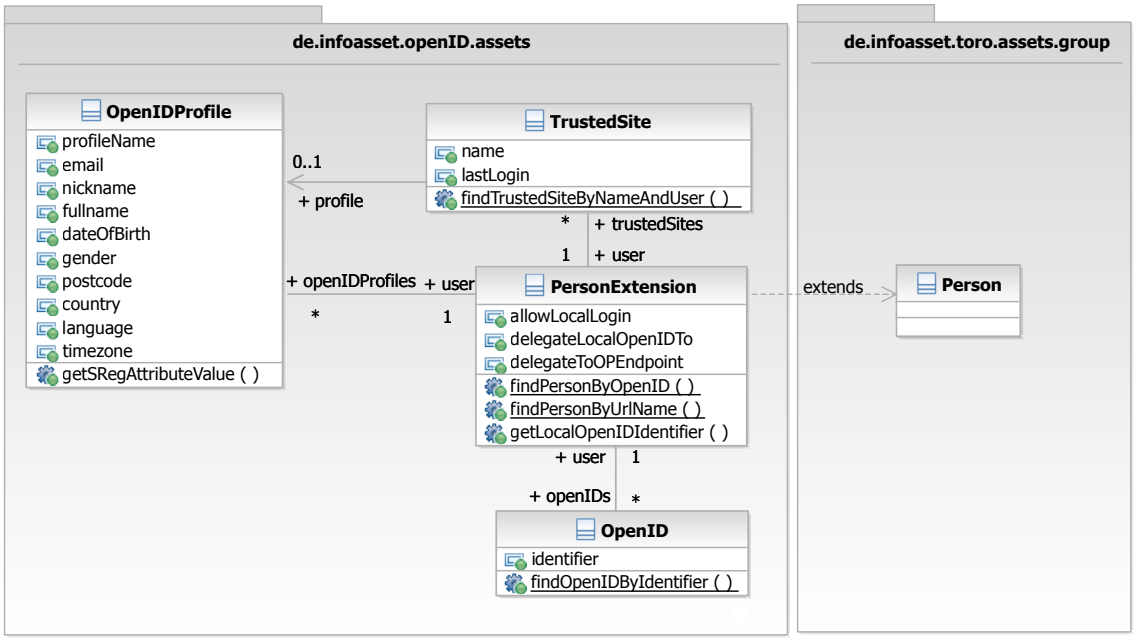


Abbildung 12.1.: Assets des OpenIDPlugins

12.1.1. PersonExtension

Kern des `de.infoasset.openid.assets` Packages bildet die *PersonExtension* Klasse, die das bestehende *Person* Asset des Toro Plugins um folgende Eigenschaften und Methoden erweitert:

- *allowLocalLogin*: Boolescher Wert, der festlegt, ob der Benutzer den lokalen passwortbasierten Anmeldemechanismus nutzen darf oder nicht.

- *delegateLocalOpenIDTo*: Speichert die OpenID zu der alle Anfragen auf die lokale OpenID des Benutzers weitergeleitet werden sollen.
- *delegateToOPEndpoint*: Speichert die URL des OP der OpenID, auf die die lokale OpenID weitergeleitet werden soll.
- *findPersonByOpenID()*: Ermittelt anhand einer OpenID den verknüpften Benutzer.
- *findPersonByUrlName()*: Ermittelt einen Benutzer anhand seines URL formatierten Namens.
- *getLocalOpenIDIdentifier()*: Liefert die lokale OpenID des Benutzers.

Zusätzlich führt die *PersonExtension* noch Verbindungen zu anderen Asstes ein: Einem erweiterten Person Asset können beliebig viele *OpenID*, *OpenIDProfile* und *TrustedSite* Assets zugeordnet werden.

12.1.2. OpenID

Das *OpenID* Asset dient zur Speicherung von OpenIDs fremder RPs, die mit einem lokalen Benutzerkonto verknüpft sind. Das Asset stellt folgende Eigenschaften und Methoden bereit:

- *identifier*: Speichert den eindeutigen Identifier einer OpenID.
- *findOpenIDByIdentifier()*: Ermittelt anhand eines OpenID Identifiers eine OpenID.

12.1.3. OpenIDProfile

Mit dem *OpenIDProfile* Asset können die verschiedenen Profile eines Benutzers zur Datenweitergabe an fremde RP gespeichert werden:

- *profileName*: Speichert den eindeutigen Namen des Profils.
- Über die optionale Eigenschaften *email*, *nickname*, *fullname*, *dateOfBirth*, *gender*, *postcode*, *country*, *language* und *timezone* können die Profildaten festgelegt werden (siehe dazu Abschnitt 10.2).
- *getSRegAttributeValue()*: Liefert den formatierten Wert eines Profilattributs, so dass er die Spezifikationen der OpenID Simple Registration Extension (siehe Abschnitt 5.2.1) erfüllt.

12.1.4. TrustedSite

Das *TrustedSite* Asset dient zur Speicherung von Seiten, denen ein Benutzers vertraut:

- *name*: Speichert den Namen der Seite.
- *lastLogin*: Speichert den Zeitpunkt der letzten Authentifizierung des Benutzers für die betroffene Seite.
- *findTrustedSiteByNameAndUser()*: Ermittelt eine Seite, die ein bestimmter Benutzer als vertrauenswürdig eingestuft hat, anhand ihres Namens.

12.2. Handler

Für die Verarbeitung der HTTP Anfragen stellt das OpenIDPlugin insgesamt 22 Handler bereit, die in Abbildung 12.2 grafisch dargestellt sind. Alle Handler des OpenID Plugins sind in dem Package `de.infoasset.openid.handler.openid` enthalten, das sich wiederum in weitere Packages unterteilt.

12.2.1. Handler für den Betrieb einer Relying Party

LoginHandler

Der *LoginHandler* stellt ein Formular zur Eingabe einer OpenID bereit (siehe Abbildung 12.3). Schickt der Benutzer das Formular ab, wird das Formular vom *DiscoveryHandler* ausgewertet. Dieser Vorgang entspricht Schritt 1 aus Abbildung 5.1.

DiscoveryHandler

Der *DiscoveryHandler* ist für die Ausführung von OpenID Anfragen an fremde OP zuständig. Der Ablauf des *DiscoveryHandlers* ist in Abbildung 12.4 grafisch dargestellt:

1. Die als Parameter übergebene OpenID wird geparst und ausgewertet. Dieser Schritt entspricht Schritt 2 und 3 in Abbildung 5.1.

2. Es wird geprüft, ob der Benutzer sich mit der OpenID registrieren möchte oder ob er versucht sich mit einer OpenID anzumelden, die noch nicht mit einem lokalen Benutzerkonto verknüpft wurde. Ist dies der Fall, wird eine Attributanfrage für den vollen Namen und die E-Mail Adresse generiert.

3. Anschließend wird eine OpenID Authentifizierungsanfrage anhand der aus Schritt 1 gewonnen Informationen erzeugt. In der Authentifizierungsanfrage wird hinterlegt, dass die Antwort auf diese Anfrage an die URL des *VerifyHandlers* gesendet werden soll. Wurde in Schritt 2 eine Attributanfrage erzeugt, wird diese in Authentifizierungsanfrage eingebettet.

4. Im letzten Schritt wird die erzeugte Anfrage über eine Browserweiterleitung an den in Schritt 1 ermittelten OP geschickt. Dieser Schritt entspricht Schritt 4 in Abbildung 5.1.

VerifyHandler

Der *VerifyHandler* nimmt die Antworten eines OPs auf OpenID Authentifizierungsanfragen entgegen, die zuvor vom *DiscoveryHandler* an den OP gestellt wurden. Die einzelnen Schritte dieser Auswertung sind in Abbildung 12.5 grafisch dargestellt:

1. Die Antwort des OP wird ausgewertet. Dieser Schritt entspricht Schritt 7 in Abbildung 5.1.

2. Es wird geprüft, ob die OpenID vom OP nicht verifiziert wurde. Ist dies nicht der Fall, ist der Vorgang abgeschlossen und der *VerifyHandler* leitet den Benutzer mit einer Fehlermeldung zurück zu dem Formular, über das er die OpenID eingegeben hat

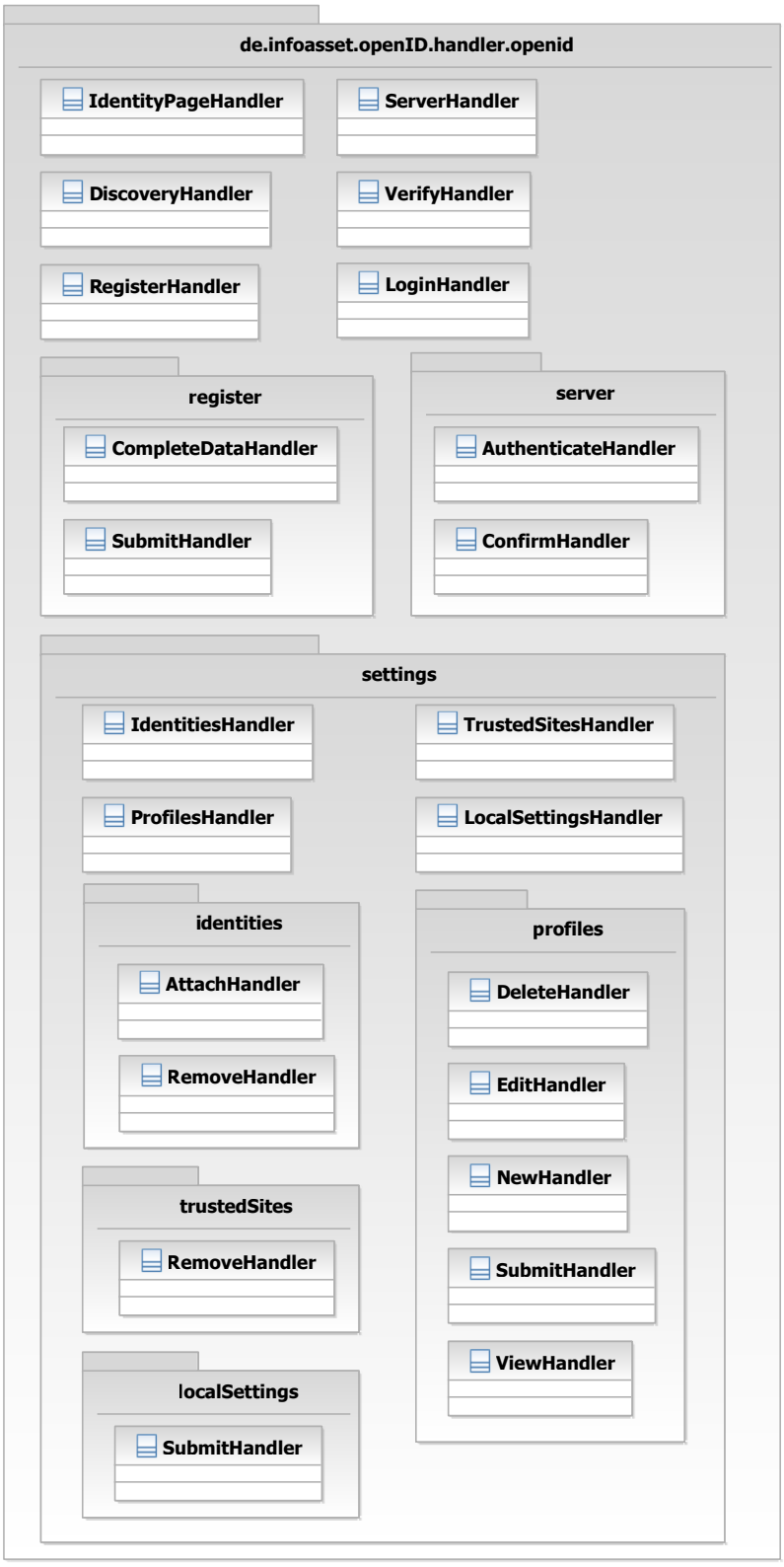


Abbildung 12.2.: Übersicht aller Handler des OpenID Plugins

Abbildung 12.3.: Formular zur Eingabe einer OpenID

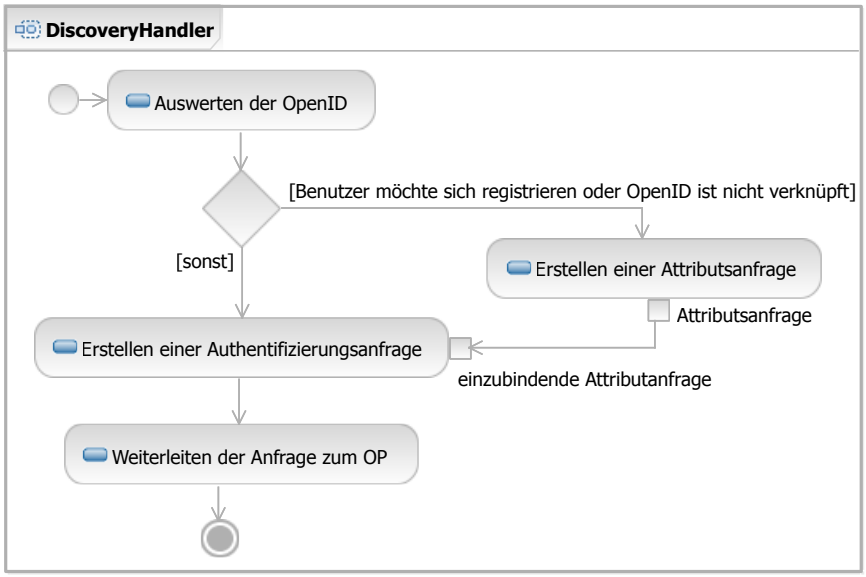


Abbildung 12.4.: Ablauf des DiscoveryHandlers

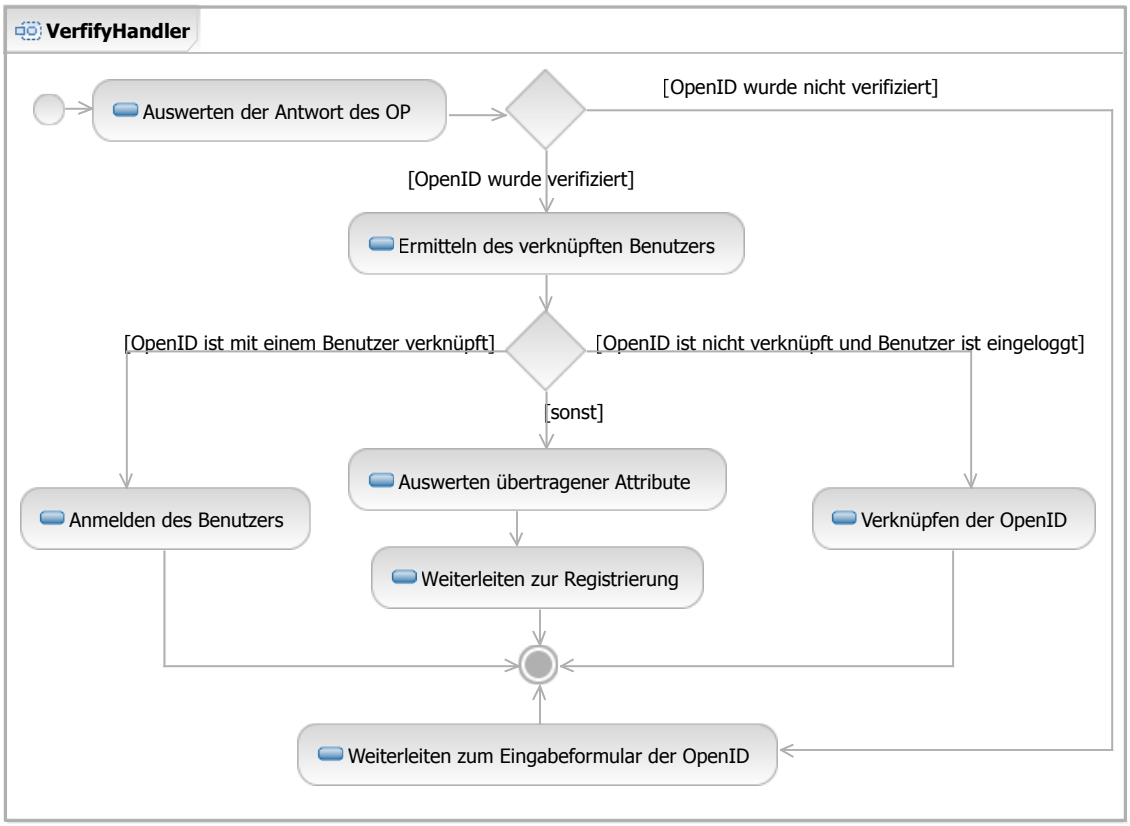


Abbildung 12.5.: Ablauf des VerifyHandler

(*LoginHandler*, *RegisterHandler* oder *AttachHandler* aus dem `de.infoasset.openid.handler.openid.settings.identities` Package (siehe Abschnitt 12.2.4)).

3. Anschließend wird geprüft, ob der Benutzer angemeldet ist und ob die OpenID bereits mit einem lokalen Benutzerkonto verknüpft ist:
   - Ist die OpenID noch nicht verknüpft und der Benutzer angemeldet, wird die OpenID mit dem Konto des angemeldeten Benutzers verknüpft.
   - Ist die OpenID mit einem Benutzerkonto verknüpft, wird der Benutzer mit diesem Konto angemeldet.
   - Tritt keiner der beiden Fälle ein, werden eventuell übertragende Attribute aus der Antwort des OP ausgewertet. Anschließend wird der Benutzer mit diesen Daten zur Registrierung (siehe Abschnitt Handler für die OpenID basierte Registrierung) weitergeleitet.

12.2.2. Handler für die OpenID basierte Registrierung

Im `de.infoasset.openid.handler.openid.register` Package sind zwei Handler zum Abschluss einer OpenID basierten Registrierung enthalten:

SubmitHandler

Möchte ein Benutzer sich per OpenID registrieren wird er nach der Eingabe und Verifikation seiner OpenID vom *VerifyHandler* zum *SubmitHandler* weitergeleitet. Dieser prüft die erhaltenen Daten auf ihre Gültigkeit. Sind die Daten ungültig oder unvollständig, wird der Benutzer zum *CompleteDataHandler* weitergeleitet. Liegen gültige Daten vor, legt der *SubmitHandler* ein neues Benutzerkonto an (vergleiche Abschnitt 8.2.1). Die OpenID des Benutzer wird dabei automatisch mit dem neuen Konto verknüpft. Da der Benutzer bei der OpenID basierten Registrierung kein Passwort angeben muss, wird die *allowLocalLogin* Eigenschaft des Benutzerkontos (siehe Abschnitt 12.1.1) auf *false* gesetzt, so dass der Benutzer sich vorerst nur mit seiner OpenID beim System anmelden kann. Der Benutzer kann diese Einstellung jederzeit ändern und sich ein lokales Passwort generieren lassen mit dem er dann auch den lokalen Anmeldemechanismus nutzen kann.

CompleteDataHandler

Der *CompleteDataHandler* stellt dem Benutzer ein Formular zur Korrektur seiner Registrierungsdaten zur Verfügung. Schickt der Benutzer das Formular ab, wird er wieder zum *SubmitHandler* geleitet.

12.2.3. Handler für den Betrieb eines OpenID Providers

IdentityPageHandler

Der *IdentityPageHandler* ist für die Ausgabe der *OpenID Identity Page* der registrierten Benutzer zuständig. Er wird ausgeführt, wenn die URL einer lokalen OpenID aufgerufen wird. Hat der Benutzer eine persönliche Beschreibung hinterlegt, wird diese Beschreibung auf seiner OpenID Identity Page angezeigt. Liegt keine persönliche Beschreibung vor, wird die in Abbildung 12.6 dargestellte Standardseite angezeigt.

Abbildung 12.6.: Standard OpenID Identity Page eines Benutzers

Neben der Präsentation des Benutzers hat die OpenID Identity Page vor allem aber auch eine technische Funktion: Sie enthält die nötigen Informationen zur Kontaktierung des OP, der für die Verwaltung der OpenID zuständig ist. Dazu werden im HTML-Kopf der Seite über zwei `<link>` Tags die URL des OP bekannt gegeben:

```
<link rel="openid2.provider" href="$serverUrl$" />
<link rel="openid.server" href="$serverUrl$" />
```

Der Platzhalter *$serverUrl$* wird durch die URL ersetzt, über die fremde RPs OpenID Anfragen an das lokale System senden können.

Hat der Benutzer eine Weiterleitung seiner lokalen OpenID eingestellt, wird der Platzhalter *$serverUrl$* durch die URL des OP ersetzt, der die OpenID verwaltet, auf die die lokale OpenID weitergeleitet werden soll. Zusätzlich wird die Ziel-OpenID noch über folgende `<link>` Tags im HTML-Kopf der OpenID Identity Page bekannt gegeben:

```
<link rel="openid2.local_id" href="$delegateIdentifier$"/>
<link rel="openid.delegate" href="$delegateIdentifier$"/>
```

ServerHandler

Der *ServerHandler* verarbeitet und beantwortet OpenID Anfragen, die fremde RPs an das System stellen. In Abbildung 12.7 ist der Ablauf der Anfrageverarbeitung grafisch dargestellt. Bei jedem Aufruf des *ServerHandlers* findet zu erst eine Auswertung der Anfrage statt, bei der alle nötigen Parameter ermittelt werden. Je nachdem ob es sich um eine *associate*, *check_authentication*, *checkid_setup* oder *checkid_immediate* Anfrage (siehe Abschnitt 5.1.2) handelt variieren die nachfolgenden Schritte.

Bei der Bearbeitung einer *associate* oder *check_authentication* Anfragen wird mithilfe des *ServerManagers* aus der OpenID4Java Biblitothek (siehe 11.2.1) die passenden Antwort generiert und direkt an die RP gesendet.

Anfragen vom Typ *checkid_setup* oder *checkid_immediate* erfordern hingegen eine mehrstufige Bearbeitung:

1. Hat der Benutzer die Anfrage in einem vorherigen Schritt bereits abgelehnt oder den Vorgang abgebrochen, ist die Bearbeitung abgeschlossen und der RP wird eine negative Antwort gesendet, alternativ wird mit dem nächsten Schritt fortgefahren.

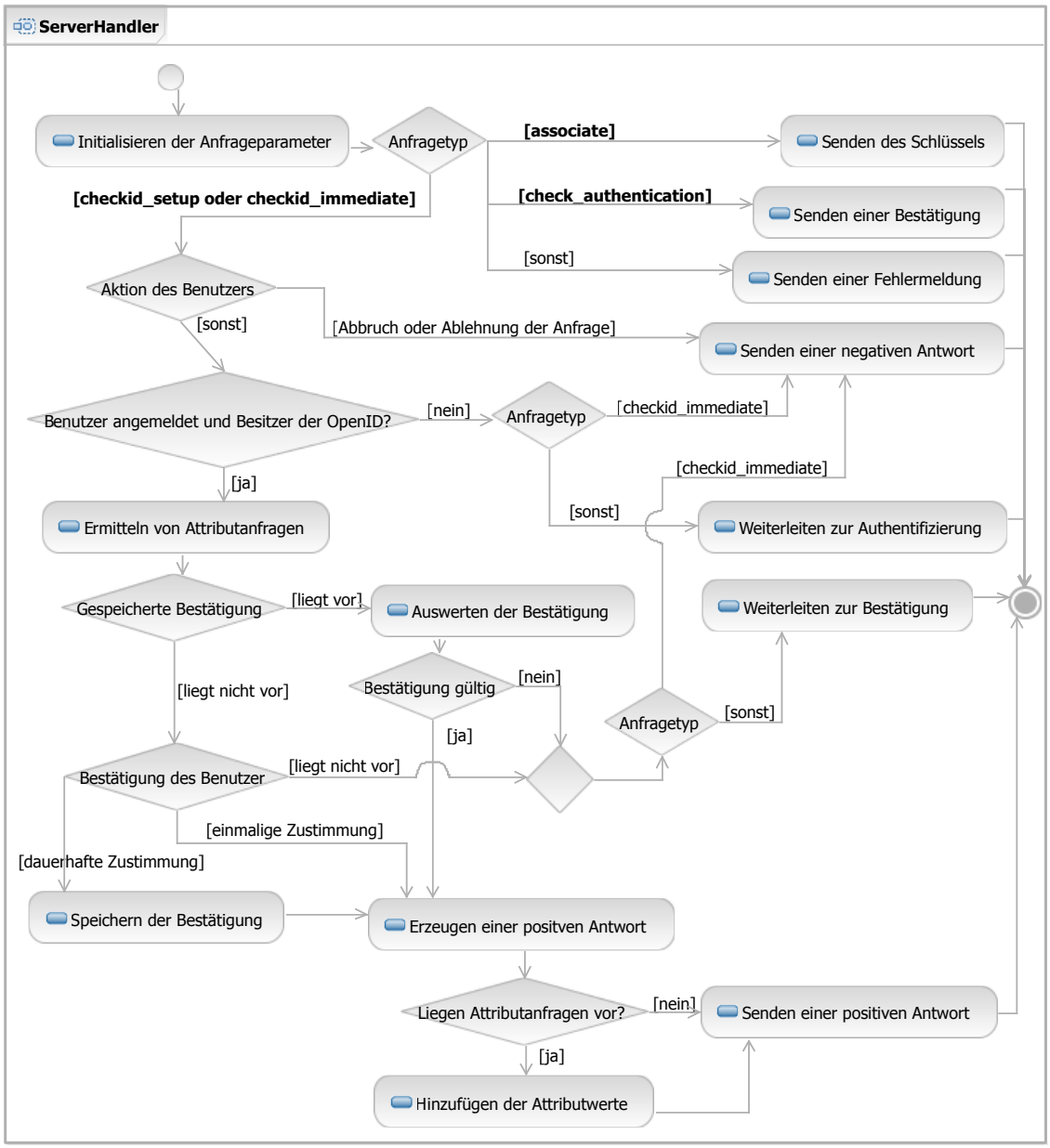


Abbildung 12.7.: Ablauf des ServerHandler

2. Es wird geprüft, ob der Benutzer bereits am System angemeldet ist. Ist der Benutzer angemeldet, wird mit dem nächsten Schritt fortgefahren, ansonsten wird die Bearbeitung je nach Anfragetyp wie folgt abgeschlossen:
   - *checkid_immediate*: Der RP wird eine negative Antwort gesendet, da eine Authentifizierung ohne die Interaktion mit dem Benutzer nicht möglich ist.
   - *checkid_setup*: Der Benutzer wird zum *AuthenticateHandler* aus dem `de.infoasset.openid.handler.openid.server` Package (siehe Abschnitt 12.2.3) weitergeleitet um in dort authentifizieren zu können.
3. Es wird analysiert, ob in der Anfrage eine Attributanfrage enthalten ist.
4. Es wird geprüft, ob eine gespeicherte Bestätigung für die Anfrage vorliegt beziehungsweise ob der Benutzer in der Vergangenheit die anfragende RP als vertrauenswürdige Seite eingestuft hat:
   - Liegt keine gespeicherte Bestätigung vor wird je nach Anfragetyp wie folgt fortgefahren:
     - *checkid_immediate*: Die Bearbeitung der Anfrage wird durch das Senden einer negative Antwort an die RP abgeschlossen, da eine Authentifizierung ohne die Interaktion mit dem Benutzer nicht möglich ist.
     - *checkid_setup*: Es wird mit Schritt 5 fortgefahren.
   - Liegt eine gespeicherte Bestätigung vor, wird geprüft, ob diese Bestätigung für die vorliegende Anfrage verwendet werden kann:
     - Liegt keine Attributanfrage vor oder wurde in der Vergangenheit beim Speichern der dauerhaften Genehmigung ein Profil an die RP gesendet, kann eine positive Antwort generiert werden. Im Falle einer Attributanfrage werden die Daten des gespeicherten Profils in die Antwort aufgenommen. Die Bearbeitung der Anfrage wird durch den Versand der positiven Antwort an die RP abgeschlossen.
     - Liegt eine Attributanfrage vor aber beim Speichern der dauerhaften Genehmigung in der Vergangenheit wurde kein Profil an die RP gesendet, wird je nach Anfragetyp wie folgt fortgefahren:
       - *checkid_immediate*: Die Bearbeitung der Anfrage wird durch das Senden einer negative Antwort an die RP abgeschlossen, da eine Authentifizierung ohne die Interaktion mit dem Benutzer nicht möglich ist.
       - *checkid_setup*: der Benutzer wird zum *ConfirmHandler* aus dem `de.infoasset.openid.handler.openid.server` Package (siehe Abschnitt 12.2.3) weitergeleitet, um dort ein Profil auswählen zu können.
5. Wurde der Benutzer nicht vom *ConfirmHandler* aus dem `de.infoasset.openid.handler.openid.server` Package (siehe Abschnitt 12.2.3) zum *ServerHandler* weitergeleitet, wird der Benutzer zum *ConfirmHandler* geleitet, um dort die vorliegende Anfrage bestätigen zu können. Ansonsten wird das Ergebnis des *ConfirmHandlers* ausgewertet:
   - Liegt eine einmalige Bestätigung der Anfrage vor, wird mit Schritt 6 fortgefahren.

- Liegt eine dauerhafte Bestätigung der Anfrage vor, wird die Bestätigung gespeichert und die anfragende RP als vertrauenswürdige Seite eingestuft. Hat der Benutzer ein Profil zur Datenweitergabe an die RP ausgewählt, wird eine Verknüpfung zu diesem Profil zusammen mit der Bestätigung gespeichert. Anschließend wird mit dem nächsten Schritt fortgefahren.

6. Eine positive Antwort wird erstellt und im Falle einer Attributanfrage werden die Daten des gespeicherten Profils in die Antwort aufgenommen. Die Bearbeitung der Anfrage wird durch den Versand der positiven Antwort an die RP abgeschlossen.

Im `de.infoasset.openid.handler.openid.server` Package befinden sich Handler die vom *ServerHandler* zur Interaktion mit dem Benutzer während der Verarbeitung einer Authentifizierungsanfrage genutzt werden können.

AuthenticateHandler

Damit eine Authentifizierungsanfrage erfolgreich beantwortet werden, kann ist es erforderlich, dass der Benutzer und Besitzer der OpenID beim System angemeldet ist. Ist dies nicht der Fall und die anfragende RP erlaubt die Interaktion mit dem Benutzer, leitet der *ServerHandler* den Benutzer zum *AuthenticateHandler* mit der Aufforderung sich anzumelden (siehe Abbildung 12.8). Nach einer erfolgreichen Authentifizierung wird der Benutzer wieder zum *ServerHandler* geleitet, damit dieser mit der Verarbeitung der Anfrage fortfahren kann.

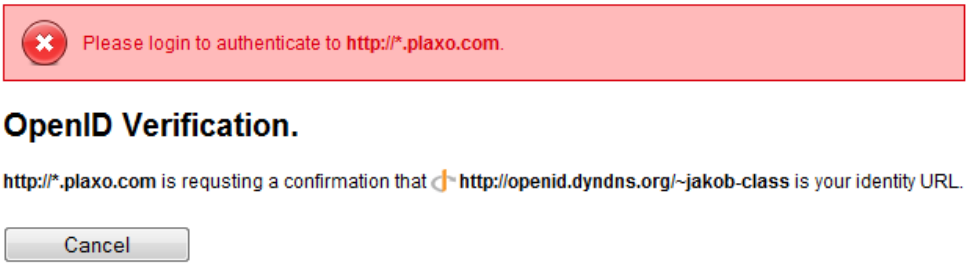


Abbildung 12.8.: Maske mit Hinweisen zur Anfrage und einer Aufforderung zur Anmeldung

ConfirmHandler

Hat der *ServerHandler* keine passende gespeicherte Bestätigung für eine Authentifizierungsanfrage finden können, leitet dieser den Benutzer zum *ConfirmHandler*. Der *ConfirmHandler* bietet dem Benutzer die Möglichkeit eine Anfrage abzulehnen, einmalig oder dauerhaft zu bestätigen. Sofern eine Attributanfrage vorliegt, kann der Benutzer zusätzlich ein Profil auswählen, editieren oder ein neues Profil anlegen, das an die anfragende RP gesendet wird (siehe Abbildung 12.9).

12.2.4. Handler für die Verwaltung persönlicher OpenID Einstellungen

Das `de.infoasset.openid.handler.openid.settings` Package enthält die Handler über die ein Benutzer seine persönlichen OpenID Einstellungen verwalten kann. Ein angemeldeter

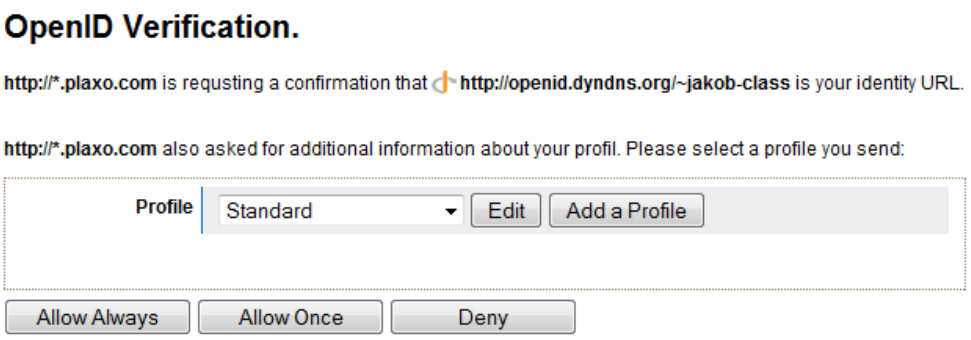


Abbildung 12.9.: Formular zur Bestätigung einer OpenID Anfrage

Benutzer gelangt durch Klicken auf seinen Namen im oberen Bereich der Webseite (siehe Abbildung 12.10) zur Verwaltung seines Kontos.

Abbildung 12.10.: Link zur Verwaltung des persönlichen Benutzerkontos

Dort stellt das OpenID Plugin dem Benutzer zusätzlich folgende Konfigurationsmöglichkeiten bereit:

- Lokale OpenID Einstellungen (*LocalSettingsHandler*)
- Verknüpfte OpenIDs (*IdentitiesHandler*)
- OpenID Profile (*ProfilesHandler*)
- Vertrauenswürdige Seiten (*TrustedSitesHandler*)

LocalSettingsHandler

Der *LocalSettingsHandler* stellt dem Benutzer das in Abbildung 12.11 dargestellte Formular zur Konfiguration seiner lokalen OpenID Einstellungen zur Verfügung. Schickt der Benutzer das Formular ab, werden die eingebenden Daten vom *SubmitHandler* aus dem `de.infoasset.openid.handler.openid.settings.localSettings` Package nach einer erfolgreichen Prüfung auf ihre Gültigkeit abgespeichert.

IdentitiesHandler

Der *IdentitiesHandler* listet alle OpenIDs, die mit dem Konto des angemeldeten Benutzers verknüpft sind, wie in Abbildung 12.12 dargestellt tabellarisch auf.

Der Benutzer hat die Möglichkeit eine Verknüpfung zwischen seinem Konto und einer OpenID zu löschen, indem er den Button zum Löschen der jeweiligen Verknüpfung in der rechten Aktionsspalte betätigt. Nach einer Bestätigung der Löschaktion wird die Verknüpfung vom *RemoveHandler* aus dem `de.infoasset.openid.handler.openid.settings.identities` Package entfernt.

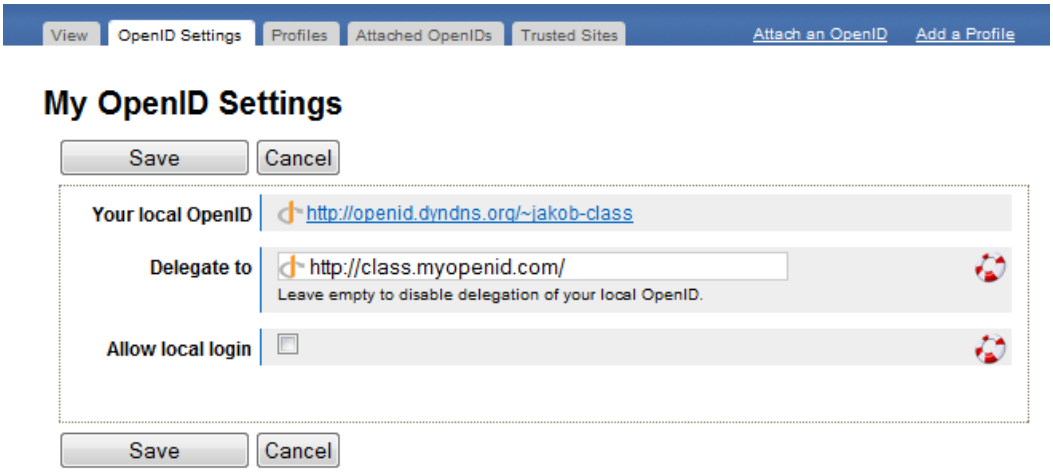


Abbildung 12.11.: Formular zur Verwaltung der lokalen OpenID Einstellungen

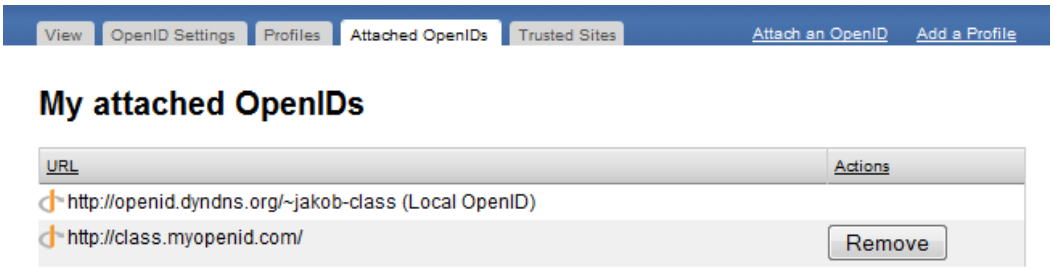


Abbildung 12.12.: Maske zur Verwaltung der verknüpften OpenIDs

Über den „*Attach an OpenID*“ Link auf der rechten Seite im oberen Anzeigebereich (siehe Abbildung 12.12) kann der Benutzer eine zusätzliche OpenID mit seinem Konto verknüpfen. Der Link führt zum *AttachHandler* aus dem `de.infoasset.openid.handler.openid.settings.identities` Package. Dieser stellt dem Benutzer ein Formular zur Eingabe der OpenID bereit (siehe Abbildung 12.3).

ProfilesHandler

Über den *ProfilesHandler* kann der Benutzer seine OpenID Profile verwalten. Der Handler liefert dem Benutzer eine Maske, auf der alle gespeicherten Profile des Benutzers aufgelistet sind (siehe Abbildung 12.13).

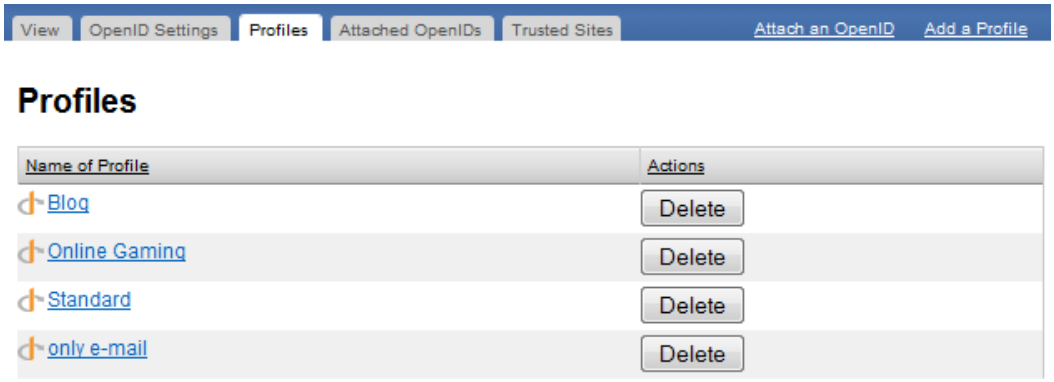


Abbildung 12.13.: Maske zur Verwaltung der OpenID Profile

Für die einzelnen Verwaltungsaktionen existieren im `de.infoasset.openid.handler.openid.settings.profiles` Package folgende Handler:

- *NewHandler*: Stellt dem Benutzer das in Abbildung 12.14 dargestellte Formular zur Erstellung eines neuen Profils zur Verfügung. Schickt der Benutzer das Formular ab wird er zum *SubmitHandler* weitergeleitet.
- *ViewHandler*: Zeigt die Details eines Profils an.
- *EditHandler*: Stellt dem Benutzer das in Abbildung 12.14 dargestellte Formular zur Bearbeitung eines vorhanden Profils zur Verfügung. Schickt der Benutzer das Formular ab, wird er zum *SubmitHandler* weitergeleitet.
- *SubmitHandler*: Speichert die Daten eines Profils nach einer erfolgreichen Prüfung auf ihre Gültigkeit ab.
- *DeleteHandler*: Löscht ein vorhandenes Profil.

TrustedSitesHandler

Der *TrustedSitesHandler* listet dem Benutzer alle Seiten auf, die er in der Vergangenheit als vertrauenswürdig eingestuft hat. In der Übersicht werden neben dem Namen der Seite auch der Name des Profils, das an die Seite gesendet wurde, und der Zeitpunkt der letzten Anmeldung angezeigt.

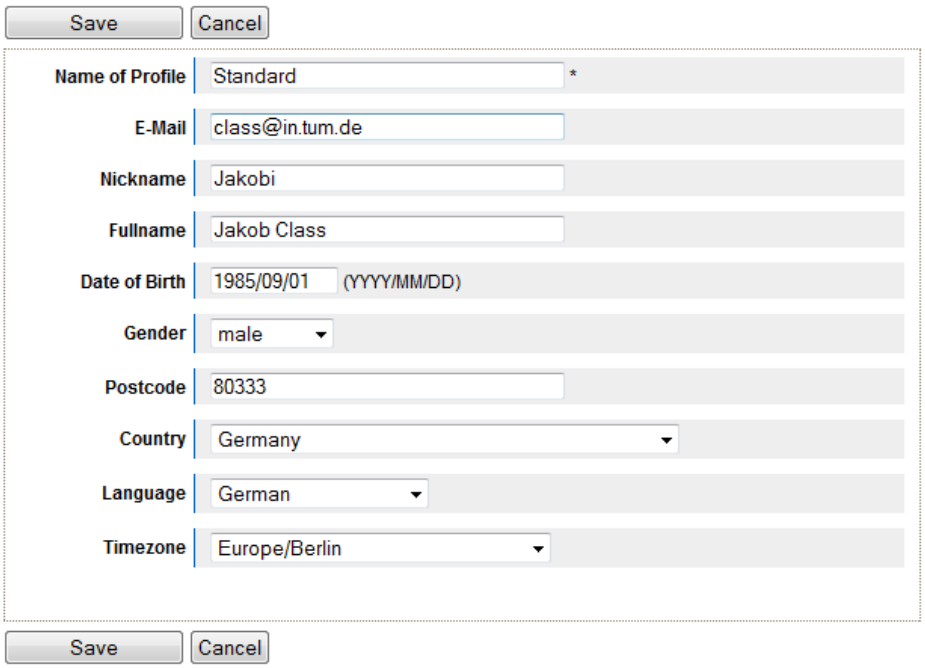


Abbildung 12.14.: Formular zur Erstellung oder Bearbeitung eines Profils

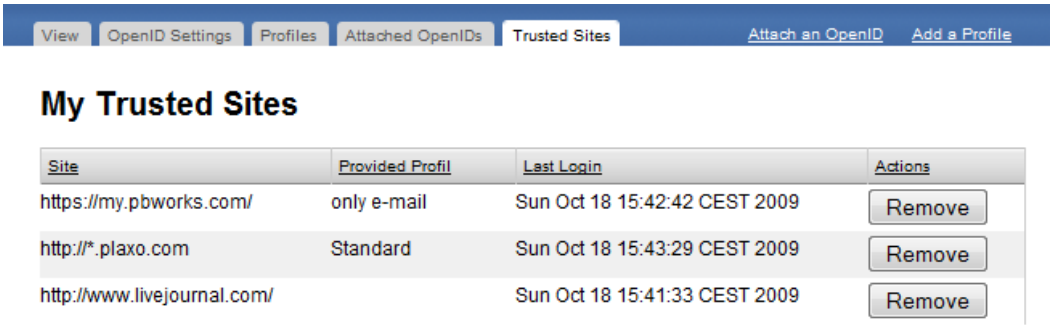


Abbildung 12.15.: Maske zur Verwaltung der vertrauenswürdigen Seiten

Über den *Löschen* Button kann der Benutzer eine Seite aus der Liste seiner vertrauenswürdigen Seiten entfernen.

13. Zusammenfassung und Ausblick

In dieser Arbeit wurde der freie SSO Standard OpenID beschrieben und analysiert. Der Standard bietet einen interessanten und ausgereiften Lösungsansatz für die Schaffung von Web SSO Umgebungen. Die in Abbildung 13.1 dargestellte Entwicklung der letzten Jahre zeigt, dass die Verbreitung von OpenID enorm steigt. Die Integration des Standards in Portale namhafter Firmen wie zum Beispiel Yahoo!, Google, AOL oder facebook fördern die Verbreitung zusätzlich.

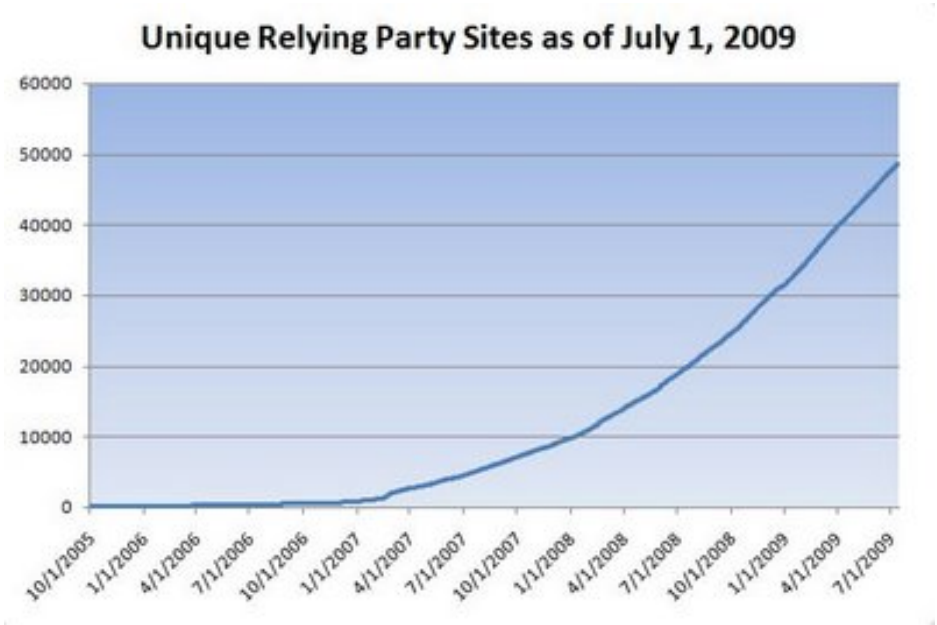


Abbildung 13.1.: Entwicklung der Anzahl von Webseiten, die OpenID unterstützen [Janc].

Allerdings gilt es beim Einsatz von OpenID die Schwachstellen und Gefahren zu beachten. Besonders die Anfälligkeit gegenüber Phishing Angriffen, durch die unberechtigte Dritte Zugriff auf sensible Daten erlangen können, stellt eine große Gefahr dar. Der Benutzer kann solche Angriffe jedoch verhindern, indem er allgemein wachsam ist. Zusätzlich kann der Einsatz von OpenID Browserplugins, die den Benutzer bei einem Authentifizierungsvorgang unterstützen, helfen, die Phishing Gefahr zu mindern.

Die Betreiber eines OP können durch die Integration Phishing resistenter Authentifizierungsmechansimen in ihre Systeme, wie zum Beispiel der Einsatz digitaler Zertifikate oder Challenge-Response Systeme, einen effektiven Beitrag gegen Phishing leisten.

Hält die momentane Entwicklung unter Berücksichtigung der potentiellen Risiken an, kann sich OpenID in den nächsten Jahren zu dem Web SSO Standard etablierten.

Die Integration von OpenID in Tricia wurde erfolgreich durchgeführt. Nach der Analyse der Anforderungen wurde ein Modell für ein Plugin entwickelt, mit dem Tricia sowohl als RP als auch als OP betrieben werden kann. Neben den Kernfunktionalitäten zum Stellen und Beantworten von Authentifizierungsanfragen bietet das Plugin zusätzliche Funktionalitäten, die den OpenID Betrieb für die Benutzer komfortabler gestalten. So können Tricia Benutzer zum Beispiel RPs als vertrauenswürdig eingestuft werden, damit Anfragen schnell und automatisch beantwortet werden können. Über die Verwaltung verschiedener Profile können die Benutzer entscheiden, welche Identitätsinformationen an andere RPs weitergegeben werden können.

Das entwickelte Modell konnte ohne größere Komplikationen umgesetzt werden. Für die Implementierung der OpenID Funktionalitäten wurde die freie OpenID4Java Bibliothek verwendet, die derzeit einen reiferen Entwicklungsstand als JOID und eine einfachere Integration als WSO2 Identity Server und NetMesh InfoGrid LID bietet. OpenID4Java erfüllt alle Ansprüche an eine OpenID Bibliothek und ist für die Integration von OpenID in Java Web-Applikationen sehr zu empfehlen.

Ein Test des OpenID Plugins in der Praxis konnte bis zum Zeitpunkt der Fertigstellung dieser Arbeit noch nicht durchgeführt werden. Allerdings konnten alle zu implementierende Anwendungsfälle in einer Testumgebung erfolgreich durchgespielt werden.

Für die Zukunft sind zahlreiche Erweiterungen des Plugins denkbar. So könnte zum Beispiel zur Steigerung der Usability das OpenID Anmeldeformular mit dem OpenID Selector [Opea] erweitert werden (siehe Abbildung 6.1 auf Seite 27). Denkbar wäre auch eine Implementierung von AX (siehe Abschnitt 5.2.2), womit komplette Benutzerkonten zwischen verschiedenen Tricia Instanzen untereinander ausgetauscht werden könnten.

Teil IV.

Anhänge

A. Anwendungsfälle

A.1. Registrierung mit einer OpenID

Tabelle A.1.: Anwendungsfall A.1

ID	A.1
Name	Registrierung mit einer OpenID.
Beschreibung	Ein Benutzer registriert ein neues lokales Benutzerkonto durch die Angabe einer OpenID.
Akteure	Benutzer, fremder OP.
Auslöser	Benutzer.
Vorbedingungen	Benutzer hat noch kein lokales Benutzerkonto, ist im Besitz einer OpenID bei dem fremden OP und hat dort seine Benutzerdaten hinterlegt. Das System vertraut dem fremden OP.
Ergebnis	Benutzer hat ein lokales Benutzerkonto, das mit seiner OpenID des fremden OP verknüpft ist und ist am System angemeldet.
Standardablauf	1 Der Benutzer ruft die Seite zur Registrierung auf. 2 Der Benutzer klickt dort auf den Link zur OpenID basierten Registrierung. 3 Das System stellt dem Benutzer ein Formular zur Eingabe seiner OpenID bereit. 4 Der Benutzer gibt seine OpenID in das Eingabefeld ein und schickt das Formular ab. 5 Das System analysiert die OpenID und ermittelt den OP der eingegebenen OpenID. 6 Das System leitet den Benutzer zum OP. 7 Der OP authentifiziert den Benutzer. 8 Der OP leitet den Benutzer zurück zum System und übergibt dabei den Namen, sowie die Email-Adresse des Benutzers an das System. 9 Das System verifiziert, ob die Daten vom OP stammen. Ist dies nicht der Fall, wird der Vorgang mit einer Fehlermeldung abgebrochen. 10.1 Das System prüft die Daten (Name und Email-Adresse) auf ihre Gültigkeit. Ist dies der Fall wird mit Schritt 12 fortgefahren. 10.2 Das System stellt dem Benutzer ein Formular zur Korrektur seiner Daten (Name und Email-Adresse) bereit. Die Felder sind mit den bereits vorhandenen Daten vorbelegt.

Fortsetzung auf der nächsten Seite

Anwendungsfall A.1 – Fortsetzung

10.3	Der Benutzer füllt das Formular aus und schickt es ab. Es wird mit Schritt 10.1 fortgefahren.
11	Das System erzeugt ein deaktiviertes Benutzerkonto mit den verifizierten Daten und verknüpft die OpenID des Benutzers mit diesem Konto.
12	Das System schickt dem Benutzer einen Link an seine Email-Adresse zum Aktivieren des Benutzerkontos.
13	Der Benutzer ruft den Link zum Aktivieren seines Kontos auf.
14	Das System aktiviert das Konto des Benutzers und erzeugt eine Sitzung für den Benutzer.

A.2. Anmeldung mit einer OpenID

Tabelle A.2.: Anwendungsfall A.2

ID	A.2
Name	Anmeldung mit einer OpenID.
Beschreibung	Ein Benutzer meldet sich mit einer OpenID beim System an, die von einem fremdem OP verwaltet wird.
Akteure	Benutzer, fremder OP.
Auslöser	Benutzer.
Vorbedingungen	Der Benutzer ist im Besitz einer OpenID bei dem fremden OP, die mit einem lokalen Benutzerkonto verknüpft ist. Das System vertraut dem fremden OP.
Ergebnis	Der Benutzer ist beim System angemeldet und hat eine gültige Sitzung.
Standardablauf	1 Der Benutzer klickt auf das OpenID-Logo für die OpenID-basierte Anmeldung. 2 Das System stellt dem Benutzer ein Formular zur Eingabe seiner OpenID zur Verfügung. 3 Der Benutzer gibt seine OpenID ein und schickt das Formular ab. 4 Das System analysiert die OpenID und ermittelt den zugehörigen OP. 5 Das System leitet den Benutzer zum OP. 6 Der OP authentifiziert den Benutzer. 7 Der OP leitet den Benutzer mit der Bestätigung der Authentifizierung zum System. 8 Das System verifiziert, ob die Daten vom OP stammen. Ist dies nicht der Fall, wird der Vorgang mit einer Fehlermeldung abgebrochen. 9 Das System ermittelt das lokale Benutzerkonto, das mit der OpenID verknüpft ist.

Fortsetzung auf der nächsten Seite

Anwendungsfall A.2 – Fortsetzung		
	10	Das System erstellt eine Sitzung für den Benutzer.

A.3. Authentifizierung

Tabelle A.3.: Anwendungsfall A.3

ID	A.3	
Name	Authentifizierung.	
Beschreibung	Ein Benutzer wird vom System für eine fremde RP authentifiziert.	
Akteure	Benutzer, fremde RP.	
Auslöser	RP.	
Vorbedingungen	Der Benutzer ist im Besitz eines lokalen Benutzerkontos. Die RP vertraut dem System.	
Ergebnis	Der Benutzer ist beim System angemeldet und hat eine gültige Sitzung. Die RP weiß, ob der Benutzer authentifiziert werden konnte.	
Standardablauf	1	Die RP leitet den Benutzer zum System und übergibt dabei die (lokale) OpenID des Benutzers an das System.
	2	Das System analysiert die OpenID und ermittelt daraus das verknüpfte lokale Benutzerkonto.
	3.1	Das System prüft, ob der Benutzer mit dem ermittelten Konto angemeldet ist. Ist dies der Fall wird mit Schritt 4 fortgefahren.
	3.2	Das System stellt dem Benutzer ein Formular zur Anmeldung bereit.
	3.3	Der Benutzer authentisiert sich beim System über den lokalen passwortbasierten Anmeldemechanismus.
	3.4	Das System authentifiziert den Benutzer. Es wird mit Schritt 3.1 fortgefahren.
	4	Das System analysiert die Anfrage der RP und informiert den Benutzer, welche RP die Authentifizierung wünscht und stellt eine Maske zur Verfügung über die der Benutzer die Anfrage ablehnen, einmalig oder dauerhaft genehmigen kann.
	5	Der Benutzer wählt eine der drei Bestätigungsmöglichkeiten.
	6.a	Lehnt der Benutzer die Anfrage ab generiert das System eine negative Bestätigung.
	6.b	Genehmigt der Benutzer die Anfrage einmalig generiert das System eine positive Bestätigung.
	6.c	Genehmigt der Benutzer die Anfrage dauerhaft speichert das System die Anfrage der RP ab und generiert eine positive Bestätigung.
	7	Das System leitet den Benutzer mit der generierten Bestätigung zur RP weiter.

A.4. Authentifizierung mit Attributanfrage

Tabelle A.4.: Anwendungsfall A.4

ID	A.4
Name	Authentifizierung mit Attributanfrage.
Beschreibung	Ein Benutzer wird vom System für eine fremde RP authentifiziert und der RP werden angeforderte Attributwerte übergeben.
Akteure	Benutzer, fremde RP.
Auslöser	RP.
Vorbedingungen	Der Benutzer ist im Besitz eines lokalen Benutzerkontos. Die RP vertraut dem System.
Ergebnis	Der Benutzer ist beim System angemeldet und hat eine gültige Sitzung. Die RP weiß, ob der Benutzer authentifiziert werden konnte und hat alle angeforderten Attributwerte erhalten, die der Benutzer zur Verfügung gestellt hat.
Standardablauf	1 Die RP leitet den Benutzer zum System und übergibt dabei die (lokale) OpenID des Benutzers und eine Liste angeforderte Attribute an das System. 2 Das System analysiert die OpenID und ermittelt daraus das verknüpfte lokale Benutzerkonto. 3.1 Das System prüft, ob der Benutzer mit dem ermittelten Konto angemeldet ist. Ist dies der Fall wird mit Schritt 4 fortgefahren. 3.2 Das System stellt dem Benutzer ein Formular zur Anmeldung bereit. 3.3 Der Benutzer authentisiert sich beim System über den lokalen passwortbasierten Anmeldemechanismus. 3.4 Das System authentifiziert den Benutzer. Es wird mit Schritt 3.1 fortgefahren. 4 Das System analysiert die Anfrage der RP und informiert den Benutzer über eine Maske, welche RP die Authentifizierung wünscht und dass diese RP zusätzliche Informationen angefordert hat. Über die Maske kann der Benutzer ein vorhandenes Profil auswählen, bearbeiten oder ein neues Profil anlegen, dessen Daten an die RP gesendet werden. Zusätzlich kann der Benutzer über die Maske die Anfrage ablehnen, einmalig oder dauerhaft genehmigen. 5 Der Benutzer selektiert ein Profil. Optional kann der Benutzer in die Anwendungsfälle „Anwendungsfall A.4“ und „Anwendungsfall A.4“ wechseln um ein neues Profil anzulegen oder das selektierte Profil zu bearbeiten. Wechselt der Benutzer in einen dieser Anwendungsfälle gelangt er nach ihrer Ausführung wieder zu diesem Schritt. 6 Der Benutzer wählt eine der drei Bestätigungsmöglichkeiten. 7.a Lehnt der Benutzer die Anfrage ab generiert das System eine negative Bestätigung.
Fortsetzung auf der nächsten Seite	

Anwendungsfall A.4 – Fortsetzung

7.b	Genehmigt der Benutzer die Anfrage einmalig belegt das System alle angeforderten Attribute mit den im selektierten Profil enthalten Daten und generiert eine positive Bestätigung.
7.c	Genehmigt der Benutzer die Anfrage dauerhaft speichert das System die Anfrage der RP und das selektierte Profil ab und belegt alle angeforderten Attribute mit den im selektierten Profil enthalten Daten und generiert.
8	Das System leitet den Benutzer mit der generierten Bestätigung zur RP weiter.

A.5. Authentifizierung ohne Benutzerinteraktion

Tabelle A.5.: Anwendungsfall A.5

ID	A.5
Name	Authentifizierung ohne Benutzerinteraktion.
Beschreibung	Ein Benutzer wird vom System für eine fremde RP authentifiziert ohne dass der Benutzer mit der RP interagieren muss.
Akteure	Benutzer (passiv), fremde RP.
Auslöser	RP.
Vorbedingungen	Benutzer ist im Besitz eines lokalen Benutzerkontos. Die RP vertraut dem System.
Ergebnis	Die RP weiß, ob der Benutzer authentifiziert werden konnte und hat alle eventuell angeforderten Attributwerte erhalten, die der Benutzer zur Verfügung gestellt hat.
Standardablauf	1 Die RP leitet den Benutzer zum System und übergibt dabei die (lokale) OpenID des Benutzers und optional eine Liste angeforderte Attribute an das System. 2 Das System analysiert die OpenID und ermittelt daraus das verknüpfte lokale Benutzerkonto. 3 Das System prüft, ob der Benutzer mit dem ermittelten Konto angemeldet ist. Ist dies nicht der Fall generiert das System eine negative Bestätigung und es wird mit Schritt 6 fortgefahren. 4 Das System analysiert die Anfrage der RP und prüft, ob die für diese Anfrage eine zuvor erstellte dauerhafte Genehmigung des Benutzer vorliegt. Fordert die RP Attributwerte an, so muss in den Daten der gespeicherten Genehmigung auch ein Verweis auf ein zuvor selektiertes Profil vorliegen. Ist dies nicht der Fall generiert das System eine negative Bestätigung und es wird mit Schritt 6 fortgefahren. 5 Das System belegt alle angeforderten Attribute mit dem im zuvor selektierten Profil enthalten Daten und generiert eine positive Bestätigung.

Fortsetzung auf der nächsten Seite

Anwendungsfall A.5 – Fortsetzung	
	6 Das System leitet den Benutzer mit der generierten Bestätigung zur RP weiter.

A.6. Verwaltung der OpenID Einstellungen

Tabelle A.6.: Anwendungsfall A.6

ID	A.6
Name	Verwaltung der OpenID Einstellungen.
Beschreibung	Ein Benutzer konfiguriert seine persönlichen OpenID-Einstellungen: • Weiterleitung der lokalen OpenID zu einer OpenID bei einem anderen OP. • Aktivierung/Deaktivierung lokalen passwortbasierten Anmeldemechanismus
Akteure	Benutzer.
Auslöser	Benutzer.
Vorbedingungen	Benutzer ist im Besitz eines lokalen Benutzerkontos mit dem er angemeldet ist.
Ergebnis	Die vom Benutzer gesetzten Einstellungen sind im System gespeichert.
Standardablauf	1 Der Benutzer ruft die Seite zur Konfiguration seiner persönlichen OpenID Einstellungen auf. 2 Das System stellt dem Benutzer ein Formular zur Verfügung über das der Benutzer angeben kann ob und an welche OpenID seine lokale OpenID weitergeleitet werden soll und über die der Benutzer den lokalen passwortbasierten Anmeldemechanismus aktivieren oder deaktivieren kann. 3 Der Benutzer setzt die gewünschten Einstellungen und schickt das Formular ab. 4 Das System prüft die Einstellungen auf ihre Gültigkeit. Sind die Einstellungen ungültig wird mit Schritt 2 fortgefahren. 5 Das System speichert die Einstellungen.

A.7. Verknüpfung einer OpenID

Tabelle A.7.: Anwendungsfall A.7

ID	A.7
Fortsetzung auf der nächsten Seite	

Anwendungsfall A.7 – Fortsetzung

Name	Verknüpfung einer OpenID.
Beschreibung	Ein Benutzer verknüpft eine OpenID mit seinem lokalen Benutzerkonto.
Akteure	Benutzer, fremder OP.
Auslöser	Benutzer.
Vorbedingungen	Benutzer hat ein lokales Benutzerkonto und ist im Besitz einer OpenID bei dem fremden OP. Das System vertraut dem fremden OP. Der Benutzer ist angemeldet und hat eine gültige Sitzung beim System.
Ergebnis	Die OpenID des Benutzers beim fremden OP ist mit dem lokalen Konto des Benutzers verknüpft.
Standardablauf	1 Der Benutzer ruft die Seite zur Verknüpfung seines Konto mit einer OpenID auf. 2 Das System stellt dem Benutzer ein Formular zur Eingabe einer OpenID bereit. 3 Der Benutzer gibt seine OpenID ein und schickt das Formular ab. 4 Das System analysiert die OpenID und ermittelt den zugehörigen OP. 5 Das System leitet den Benutzer zum OP. 6 Der OP authentifiziert den Benutzer. 7 Der OP leitet den Benutzer mit der Bestätigung der Authentifizierung zum System. 8 Das System verifiziert, ob die Daten vom OP stammen. Ist dies nicht der Fall, wird der Vorgang mit einer Fehlermeldung abgebrochen. 9 Das System verknüpft die OpenID mit dem lokalen Konto des Benutzers und sendet dem Benutzer eine Bestätigung über die erfolgreiche Verknüpfung.

A.8. Entfernen einer OpenID Verknüpfung

Tabelle A.8.: Anwendungsfall A.8

ID	A.8
Name	Entfernen einer OpenID Verknüpfung.
Beschreibung	Ein Benutzer entfernt eine Verknüpfung zwischen seinem lokalen Benutzerkonto und einer OpenID.
Akteure	Benutzer.
Auslöser	Benutzer.
Vorbedingungen	Benutzer ist im Besitz eines lokalen Benutzerkontos und ist beim System angemeldet.

Fortsetzung auf der nächsten Seite

Anwendungsfall A.8 – Fortsetzung

Ergebnis	Die zu entfernende Verknüpfung zwischen dem Benutzerkonto und der OpenID ist gelöscht.
Standardablauf	1 Der Benutzer ruft die Seite zur Bearbeitung aller verknüpften OpenIDs auf. 2 Das System ermittelt alle OpenIDs, die mit dem Konto des Benutzers verknüpft sind. 3 Das System sendet dem Benutzer eine Liste mit den verknüpften OpenIDs. 4 Der Benutzer wählt eine OpenID aus der Liste, deren Verknüpfung entfernt werden soll. 5 Das System fragt noch einer Bestätigung zur Entfernung der gewählten Verknüpfung. 6 Der Benutzer bestätigt die Entfernung. 7 Das System löscht die Verknüpfung zwischen dem lokalen Benutzerkonto und der gewählten OpenID.

A.9. Entfernen einer vertrauenswürdigen Seite

Tabelle A.9.: Anwendungsfall A.9

ID	A.9
Name	Entfernen einer vertrauenswürdigen Seite.
Beschreibung	Ein Benutzer entfernt eine Seite, die er in der Vergangenheit als vertrauenswürdig eingestuft hat.
Akteure	Benutzer
Auslöser	Benutzer
Vorbedingungen	Benutzer ist im Besitz eines lokalen Benutzerkontos, ist beim System angemeldet und hat in der Vergangenheit mindestens eine Seite als vertrauenswürdig eingestuft.
Ergebnis	Die entfernte Seite ist nicht mehr in der Menge der Seiten enthalten, denen der Benutzer vertraut.
Standardablauf	1 Der Benutzer ruft die Seite zur Bearbeitung seiner vertrauenswürdigen Seiten auf. 2 Das System ermittelt alle Seiten, die der Benutzer mit seinem angemeldeten Konto in der Vergangenheit als vertrauenswürdig eingestuft hat. 3 Das System sendet dem Benutzer eine Liste mit seinen vertrauenswürdigen Seiten. 4 Der Benutzer wählt eine Seite aus der Liste, die entfernt werden soll. 5 Das System fragt noch einer Bestätigung zur Entfernung der gewählten Seite.

Fortsetzung auf der nächsten Seite

Anwendungsfall A.9 – Fortsetzung	
	6 Der Benutzer bestätigt die Entfernung. 7 Das System entfernt die gewählte vertrauenswürdige Seite.

A.10. Anlegen eines neuen Profils

Tabelle A.10.: Anwendungsfall A.10

ID	A.10
Name	Anlegen eines neuen Profils.
Beschreibung	Ein Benutzer erstellt ein neues OpenID Profil.
Akteure	Benutzer.
Auslöser	Benutzer.
Vorbedingungen	Benutzer ist im Besitz eines lokalen Benutzerkontos und ist beim System angemeldet.
Ergebnis	Das vom Benutzer erstellte Profil gespeichert und seinem Benutzerkonto zugeordnet.
Standardablauf	1 Der Benutzer ruft die Seite zum Anlegen eines neuen Profils auf. 2 Das System stellt dem Benutzer ein Formular zur Eingabe der Profildaten bereit: • Name des Profils (erforderlich) • E-Mail (optional) • Spitzname (optional) • voller Name (optional) • Geburtsdatum (optional) • Geschlecht (optional) • Postleitzahl (optional) • Land (optional) • Sprache (optional) • Zeitzone (optional) 3 Der Benutzer füllt das Formular aus und schickt es ab. 4 Das System überprüft die Gültigkeit der eingebenden Daten. Sind die Daten nicht gültig wird mit Schritt 2 fortgefahren. 5 Das System speichert das Profil ab und verknüpft es mit dem Konto des Benutzers.

A.11. Bearbeiten eines Profils

Tabelle A.11.: Anwendungsfall A.11

ID	A.11
Name	Bearbeiten eines Profils.
Beschreibung	Ein Benutzer bearbeitet eines seiner OpenID Profile.
Akteure	Benutzer.
Auslöser	Benutzer.
Vorbedingungen	Benutzer ist im Besitz eines lokalen Benutzerkontos und ist beim System angemeldet. Das zu bearbeitende Profil ist dem Benutzerkonto des Benutzers zugeordnet.
Ergebnis	Das vom Benutzer erstellte Profil gespeichert und seinem Benutzerkonto zugeordnet.
Standardablauf	1 Der Benutzer ruft die Seite zur Bearbeitung seiner Profile auf. 2 Das System ermittelt alle Profile, die mit dem Konto des Benutzers verknüpft sind. 3 Das System sendet dem Benutzer eine Liste mit seinen Profilen. 4 Der Benutzer wählt ein Profil aus der Liste, das bearbeitet werden soll. 5 Das System stellt dem Benutzer ein Formular zur Bearbeitung der Daten des gewählten Profils bereit: • Name des Profils (erforderlich) • E-Mail (optional) • Spitzname (optional) • voller Name (optional) • Geburtsdatum (optional) • Geschlecht (optional) • Postleitzahl (optional) • Land (optional) • Sprache (optional) • Zeitzone (optional) 6 Der Benutzer bearbeitet das Formular und schickt es ab. 7 Das System überprüft die Gültigkeit der eingebenden Daten. Sind die Daten nicht gültig wird mit Schritt 5 fortgefahren. 8 Das System speichert das Profil ab.

A.12. Entfernen eines Profils

Tabelle A.12.: Anwendungsfall A.12

ID	A.12
Name	Entfernen eines Profils.
Beschreibung	Ein Benutzer entfernt eines seiner OpenID Profile.
Akteure	Benutzer.
Auslöser	Benutzer.
Vorbedingungen	Benutzer ist im Besitz eines lokalen Benutzerkontos und ist beim System angemeldet. Das zu löschende Profil ist dem Benutzerkonto des Benutzers zugeordnet.
Ergebnis	Das vom Benutzer erstellte Profil gespeichert und seinem Benutzerkonto zugeordnet.
Standardablauf	1 Der Benutzer ruft die Seite zur Bearbeitung seiner Profile auf. 2 Das System ermittelt alle Profile, die mit dem Konto des Benutzers verknüpft sind. 3 Das System sendet dem Benutzer eine Liste mit seinen Profilen. 4 Der Benutzer wählt ein Profil aus der Liste, das entfernt werden soll. 5 Das System fragt noch einer Bestätigung zur Entfernung des gewählten Profils. 6 Der Benutzer bestätigt die Entfernung. 7 Das System löscht das gewählte Profil.

Abkürzungsverzeichnis

AX
Attribute Exchange.

DAC
Discretionary Access Control.

HTTP
Hypertext Transfer Protocol.

IP
Identity Provider.

MAC
Mandatory Access Control.

MVC
Model-View-Controller.

OP
OpenID Provider.

RBAC
Role Based Access Control.

RP
Relying Party.

sebis
Software Engineering for Business Information Systems.

SP
Service Provider.

SSO
Single-Sign-On.

Abbildungsverzeichnis

Literaturverzeichnis

[BD04] Bruegge, Bernd ; Dutoit, Allen H.: *Object-Oriented Software Engineering.* Upper Saddle River : Pearson Prentice Hall, 2004. – ISBN 0–13–0471100

[Con] Congress, Library of: *ISO 639-2 Registration Authority*, http://www.loc.gov/standards/iso639-2/, Abruf: 13.11.2009

[DH08] Dinger, Jochen ; Hartenstein, Hannes: *Netzwerk- und IT-Sicherheitsmanagement.* Karlsruhe : Universitätsverlag Karlsruhe, 2008. – ISBN 978–3–86644–209–2

[Fou] Foundation, OpenID: *Libraries*, http://openid.net/developers/libraries/, Abruf: 13.11.2009

[Gmb] GmbH fun c.: *OpenID Phishing demo*, http://idtheft.fun.de/, Abruf: 08.11.2009

[Gro] Group, Network W.: *The MD4 Message-Digest Algorithm*, http://tools.ietf.org/html/rfc1320, Abruf: 13.11.2009

[HBS+07] Hardt, D. ; Bufu, J. ; Sxip Identity ; Hoyt, J. ; JanRain: *OpenID Attribute Exchange 1.0 - Final*, 05. Dezember 2007. http://openid.net/specs/openid-attribute-exchange-1_0.html, Abruf: 08.03.2009

[HDJ+06] Hoyt, J. ; Daugherty, J. ; JanRain ; Recordon, D. ; VeriSign: *OpenID Simple Registration Extension 1.0*, 30. Juni 2006. http://openid.net/specs/openid-simple-registration-extension-1_0.html, Abruf: 30.06.2006

[Inc] Inc., LiveJournal: *LiveJournal*, http://www.livejournal.com/, Abruf: 13.11.2009

[ISO] ISO: *ISO 3166 code lists*, http://www.iso.org/iso/country_codes/iso_3166_code_lists.htm, Abruf: 13.11.2009

[Jana] JanRain: *About CallVerifID*, https://www.myopenid.com/about_callverifid, Abruf: 08.11.2009

[Janb] JanRain: *myOpenID*, https://www.myopenid.com/, Abruf: 13.11.2009

[Janc] JanRain: *Relying Party Stats as of July 1, 2009*, http://blog.janrain.com/search?q=Relying+Party+Stats, Abruf: 07.11.2009

[joi] joid: *joid - Project Hosting on Google Code*, http://code.google.com/p/joid/, Abruf: 13.11.2009

[KE06] Kemper, Alfons ; Eickler, Andre: *Datenbanksysteme.* München : Oldenbourg Verlag München Wien, 2006. – ISBN 978–3–486–57690–0

[MA08] Mezler-Andelberg, Christian: *Identity Management - eine Einführung. Grundlagen, Technik, wirtschaftlicher Nutzen.* Heidelberg : dpunkt.verlag GmbH, 2008. – ISBN 978–389864–438–9

[Net] NetMesh: *NetMesh InfoGrid*, http://netmesh.us/products/netmesh_infogrid/, Abruf: 13.11.2009

[Opea] *OpenID Selector.* : *OpenID Selector*, http://code.google.com/p/openid-selector/, Abruf: 07.11.2009

[opeb] openid4java: *openid4java - Project Hosting on Google Code*, http://code.google.com/p/openid4java/, Abruf: 13.11.2009

[Ope07] OpenID developer community: *OpenID Authentication 2.0 - Final*, 05. Dezember 2007. http://openid.net/specs/openid-authentication-2_0.html, Abruf: 08.03.2009

[PBw] PBworks: *PBworks*, http://pbworks.com/, Abruf: 13.11.2009

[Reh08] Rehman, Rafeeq U.: *The OpenID Book.* Conformix Technologies Inc., 2008. – ISBN 978–0–9724031–2–2

[Sun] Sun, Twin: *Sources for Time Zone and Daylight Saving Time Data*, http://www.twinsun.com/tz/tz-link.htm, Abruf: 13.11.2009

[Ver] VeriSign: *VeriSign's OpenID SeatBelt Plugin*, https://pip.verisignlabs.com/seatbelt.do, Abruf: 08.11.2009

[Wik] Wiki, OpenID: *OpenID Phishing Brainstorm*, http://wiki.openid.net/OpenID_Phishing_Brainstorm, Abruf: 08.11.2009

[Win05] Windley, Phillip J.: *Digital Identity.* O'Reilly Media Inc., 2005. – ISBN 0–596–00878–3

[WSO] WSO2: *Identity Server by WSO2*, http://wso2.org/projects/identity, Abruf: 13.11.2009

[Yah] Yahoo!: *Yahoo! Releases OpenID Research*, http://developer.yahoo.net/blog/archives/2008/10/open_id_research.html, Abruf: 07.11.2009