# Silk Server

Setting: Eine Anwendung kennt bereits eine Menge an RDF Daten zusätzlich greift sie auf das Web of Data zu um zusätzliche Daten zu finden und gemeinsam mit den bekannten Daten zu nutzen. Da nicht sichergestellt ist, das die gefundenen Web Daten mit den initial existierenden Daten verlinkt sind, will die Applikation beim Import zusätzliche Links erzeugen.

Components

1. Die Anwendung, die auf das WOD zugreift
2. Ein RDF Store auf den die App über SPARQL/SPARUL zugreift
3. Der Silk Server auf den per http zugegriffen wird.

Use Case

1. Die App stellt Silk2 eine Matching Konfiguration zur Verfügung, die beschreibt, wie alle (also mehrere) für die Anwendung interessanten Klassen gematcht werden sollen.
2. Silk2 läd sich alle für die verschiedenen Klassen relevanten Instanzdaten aus dem Store und cacht diese (erst einmal im Speicher, später können wir über andere Lösung nachdenken).
3. Die Anwendung findet Daten im Web, die sie mit den Daten im Store verlinken will.
   1. Sie schickt diese Daten (RDF Beschreibung einer oder mehrerer Ressourcen, eine oder mehrere Klassen) per HTTP an die Silk Instanz
   2. Silk schaut welche LinkSpec gemäß ihrer dataset restriction auf die Instanzen anwendbar ist und startet entsprechende Matching Jobs gegen den Cache.
   3. Wenn eine Instanz über dem Ähnlichkeits-Threshold liegt, wird ein Link „Orginal URI owl:sameAs Existierende URI im Store“ erzeugt und für die Rückgabe an die Applikation in einem Cache gesammelt. Gegebenenfalls werden Missing Values im Cache mit Daten von der neuen Instanz ergänzt (zusatzfeature version 2)
   4. Wenn eine Instanz nicht über dem Ähnlichkeits-Threshold  liegt wird davon ausgegangen, dass die Instanz bisher unbekannt ist und daher im Store gespeichert werden soll. Es passiert folgendes: Die Instanz wird in Silk Cache aufgenommen. Es wird ein Triple in den Rückgabe-Cache geschrieben „URI der Instanz silk:matchingResult silk:UnknownInstance“
   5. Wenn alle Instanzen abgearbeitet sind wird das resultierende Triple-Set an die Anwendung zurückgegeben.
4. Die Anwendung kann jetzt neue Instanzen in den RDF Store schreiben, falls sie das will bzw. die bekannten Daten im Store mit zusätzlichen, bisher fehlenden Property-Values aus den neuen Daten ergänzen, falls Sie das will.

## Requirements

### Server

1. The Server can be accessed through a REST interface.
2. It provides a interface to commit new RDF Data and to retrieve the matching result.

### Instance Cache

1. Instances are cached in memory. New instances can be added to the cache, but random access is not required.
2. A user-defined function determines if an instance, which according to the matching is not yet known, is written to the cache

### Matching Algorithm

1. Accepts one Configuration which may consist of multiple LinkSpecs
2. Incoming instances must be allocated to the correct LinkSpec based on its restriction. (A LinkSpec may contain restrictions consisting of arbitrary SPARQL patterns (not restricted to rdf:type))

## Activity Diagram



### Notes:

* The Silk.Load and Silk.Match actions are provided by Silk 2.0
* A new InstanceCache must be implemented which holds all Instances in Memory and supports adding new instances at runtime.
* A separate instance cache is used for each LinkSpec to improve the matching performance and to allow different blocking specifications.

## Possible Extensions

* Distributing the instance cache as well as the matching task on multiple machines
* Merging duplicated instances in the cache