Bachelorarbeit

Bayerische Vermessungsverwaltung

Fakultät für Geoinformation | HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN · FH MÜNCHEN

Bachelorarbeit

Entwicklung automatisierter Tests zur Qualitätssicherung von WMS- und WMTS-Diensten

Christina Müller-Wilke

Studiengang: Geoinformatik und Satellitenpositionierung

Martikelnummer: 03472712

Geburtsdatum: 13. März 1991

Betreuer: Prof. Dr. Georg Lother

Sommersemester 2016

Abgabetermin: 08.07.2016

Die Arbeit wurde in Kooperation mit dem Landesamt für Digitalisierung, Breitband und Vermessung angefertigt.

„Durch Testen kann man die Anwesenheit, nie die Abwesenheit von Fehlern zeigen.“

Edsger Wybe Dijkstra
niederländischer Informatiker

Zusammenfassung

Die vorliegende Arbeit beschäftigt sich mit der Erstellung und Umsetzung eines Konzepts zur Testautomatisierung von Web Map Services und Web Map Tile Services des Landesamtes für Digitalisierung, Breitband und Vermessung.

Die Automatisierung von Tests ist in der heutigen Zeit sehr gebräuchlich. Jedoch gestaltet sich die Erstellung solcher Tests als schwierig, da es in der Praxis keine konkreten Vorgaben oder Normen gibt, an denen man sich orientieren kann. Diese Standards müssen von der jeweiligen Institution definiert werden.

Des Weiteren besteht das Problem, dass viele Leute die Testautomatisierung häufig als günstige Alternative zu menschlichen Testern ansehen. Jedoch kann eine Maschine nie so testen wie ein Mensch. Die Automatisierung soll lediglich dazu dienen, die Tester zu unterstützen und zu entlasten. Das Konzept der von mir entwickelten automatisierten Tests bezieht nach wie vor die Tester mit ein. Nur werden von diesen jetzt die Funktionen geprüft, für die eine Automatisierung nicht rentable wäre.

Der Test beruht auf einem Soll-Ist-Vergleich. Das Capabilities-Dokument des Dienstes wird mit der jeweiligen Produktspezifikation verglichen. Diese Spezifikationen werden vom Auftraggeber erstellt. Die Grundlage des Tests ist die OWSLib. Eine Open-Source Python-Bibliothek. Sie wurde dazu entwickelt um die Informationen aus einem Capabilities-Dokument auszulesen und anzuzeigen.

Momentan wird nur das Capabilities-Dokument überprüft. Eine Automatisierung der Tests zur Prüfung des Inhalts der GetMap- und GetFeatureInfo-Abfrage ist noch in der Entwicklung.

Bei der Entwicklung von automatisierten Tests darf nicht vergessen werden, dass diese auch ständig gepflegt werden müssen. Nur wenn sie regelmäßig aktualisiert werden, kann langfristig mit ihnen gearbeitet werden

Vorwort

Schon meine ersten Unterrichtsstunden in der 7. Klasse weckten in mir ein großes Interesse an der Informatik und der Softwareentwicklung. Dieses konnte ich während meines Studiums in der Geoinformatik weiter vertiefen. Deswegen habe ich gerne das Thema Testautomatisierung von OGC-Diensten für meine Bachelorarbeit ausgewählt. Zudem handelt es sich hierbei auch um ein sehr aktuelles und spannendes Thema.

An dieser Stelle möchte ich mich ganz herzlich bei allen Personen bedanken, die mir dabei geholfen haben, diese Arbeit fertig zustellen.

Zu besonderem Dank bin ich meinem Sachgebiet 461 des LDBV verpflichtet, das sie mir die Möglichkeit und auch das Vertrauen geschenkt haben diese Tests zu entwickeln. Des Weiteren bedanke ich mich bei Frau Kerstin Breitlow und Herrn Jürgen Weichand, einmal für die Idee der automatisierten Tests und dann für die Beantwortung meiner zahlreichen Fragen. Auch bei Frau Astrid Feichtner und Frau Michaela Jud von der Geschäftsstelle der GDI-BY und bei Frau Yvonne Clerico möchte ich mich bedanken für die Beantwortung meiner Fragen.

Herrn Prof. Dr. Lother von der Hochschule für angewandte Wissenschaften in München Danke ich dafür, das er diese Arbeit betreut hat und auf jede meiner Fragen immer eine Antwort hatte.

Christina Müller-Wilke

München, am 03.07.2016

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Um die qualitativen Standards einer Software zu erstellen oder zu erhalten, ist eine intensive Prüfung vor der Freigabe unausweichlich. Jedoch ist eine ausreichende Prüfung der Software nur möglich, wenn die jeweiligen Tester genügend geschult sind und ihre Rolle und Aufgabe auch ernst nehmen. Ist dies nicht der Fall, können fehlerbehaftete Programme auf den Markt kommen, trotz Test- und Freigabeverfahren. Es kann aber auch vorkommen, dass ein Tester einen Fehler übersieht oder aber die Teststellung nicht versteht und deswegen „falsch“ testet.

Während meines Praxissemesters im Sachgebiet 461 „Qualitätsmanagement in der BVV“ (QM-IuK) im „Landesamt für Digitalisierung, Breitband und Vermessung“ (LDBV) bekam ich selbst mit, dass die Testverfahren von Web Map Services (WMS) und Web Map Tile Services (WMTS) stellenweise nicht ausreichend waren, da manche der Tester durch, zum Beispiel mangelndes Fachwissen, mehr Arbeit verursachten als Nutzen.

Unter anderem wurden deswegen die Testverfahren für WMSe und WMTSe im LDBV so gut es ging automatisiert. Dadurch wurden die Tester entlastet, da sie nicht mehr so viel Zeit für die Tests aufwenden mussten, was für QM-IuK als für den Test verantwortliche Stelle eine Erleichterung in den einzelnen Testverfahren bedeutete.

Ein weiterer Vorteil dieser Tests ist es, einzelne Abhängigkeiten zu erkennen und in seine Bestandteile zu zerlegen. Die internen Prozesse im LDBV sind zu komplex und zu unübersichtlich, als dass sie eine einzelne Person erkennen könnte. Durch die Schnelligkeit der automatisierten Testverfahren und der gespeicherten Auswertungsprotokolle kann, im Falle einer Störungs- oder Wartungsmeldung, erkannt werden, welcher Dienst zum Beispiel von welcher Datenbank abhängig ist. Dadurch werden

Schritt für Schritt die einzelnen Zusammenhänge sichtbar. Diese können dann in einem Schema aufgezeichnet werden.

Ziel dieser Bachelorarbeit ist es, das Thema Testautomatisierung im Allgemeinen vorzustellen und konkret die Entwicklung und Einführung von automatisierten Tests für WMSe und WMTSe im LDBV zu erläutern. Des Weiteren werde ich auch meine zukünftige Planung der Tests beschreiben.

1.2 Aufbau der Arbeit

In Kapitel 2 werden die fachlichen Standards der Dienste, also die Normen des Open Geospatial Consortium und der Arbeitsgemeinschaft der Vermessungsverwaltung sowie vereinzelte Standards der International Organization for Standardization erklärt. Auch wird hier auf die „fehlenden“ technischen Standards der Testautomatisierung eingegangen.

In Kapitel 3 werden die Dienste der OGC erklärt, wobei hier der Schwerpunkt auf dem Web Map Service und dem Web Map Tile Service liegt. Die einzelnen Operationen werden genau erläutert und an einem Beispiel gezeigt. Für den WMS wird als Beispiel der WMS der Digitalen Orthophotos mit 80cm Bodenauflösung (DOP80-WMS) des LDBV, für welchen auch ein Test geschrieben wurde, verwendet. Als Beispiel für den WMTS wird der BayernAtlas-WMTS des LDBV herangezogen. Der BayernAtlas-WMTS ist nicht kostenfrei und kann nur mit einem sogenannten „ServiceKey“ genutzt werden, welcher in die URL eingebunden werden muss. Deshalb wird in den Beispielen statt eines echten ServiceKey dieses Wort in geschwungenen Klammern {ServiceKey} angegeben.

Im darauffolgenden Kapitel geht es um das Testen im Allgemeinen. Hier wird erläutert was es bedeutet, eine Software zu testen und was dabei zu berücksichtigen ist. Des Weiteren wird erklärt, was es mit der Testautomatisierung auf sich hat und welche Grenzen es bei der Automatisierung gibt.

Im Fokus des 5. Kapitels stehen die automatisierten Tests. Konkret werden hier das von mir entwickelte Konzept und meine Vorgehensweise bei der Erstellung der Tests erläutert. Des Weiteren werden der momentane Zyklus derTesterstellung und die Korrektur der WMSe veranschaulicht.

Das Konzept und Vorgehen, welches in der Theorie in Kapitel 5 erklärt wird, wird im Kapitel 6 noch einmal anhand einzelner Beispiele genauer erläutert. Unter anderem wird hier verdeutlicht, worauf bei der Erstellung der Tests zu achten ist, um mögliche Fehler zu vermeiden.

In Kapitel 7 wird ein Ausblick über die Weiterentwicklungsmöglichkeiten der automatisierten Tests gegeben. Diese beziehen sich auf die Erweiterung der Tests für WMSe und WMTSe, aber auch die Aufnahme weiterer OGC-konformer Web-Dienste, sowie den Aufbau eines Monitorings.

Den Schluss der Arbeit bildet ein Fazit über die Einführung der automatisierten Tests am Landesamt für Digitalisierung, Breitband und Vermessung.

2 Standards und Normen

Im Duden ist das Wort „Norm“ in Wirtschaft, Industrie, Technik und Wissenschaft wie folgt definiert: „Vorschrift, Regel, Richtlinie oder ähnliches für die Herstellung von Produkten, die Durchführung von Verfahren, die Anwendung von Fachtermini oder ähnliches“.[1]

2.1 Technische Standards und Normen

Im Vergleich zu anderen naturwissenschaftlichen Disziplinen, steckt die Informatik und die Softwareentwicklung quasi noch in den „Kinderschuhen“. Dies betrifft nicht nur die inhaltliche Ausgestaltung, sondern auch die Durchdringung in Lehre und Praxis[2]. Ein wesentlicher Grund dafür, warum Softwareprojekte trotz in Standards manifestierter Erfahrungen selten erfolgreich sind, ist die Unverbindlichkeit von allen bekannten „best practices“[3].

Für die Entwicklung von Software und der jeweiligen Testverfahren gibt es viele verschiedene Herangehensweisen. Mit dem Wasserfallmodell, SCRUM, V-Modell oder Behavior Driven Development seien hier nur ein paar Methoden erwähnt. Durch diese vielen unterschiedlichen Möglichkeiten obliegt es jedem Unternehmen bzw. jedem Entwickler selbst zu entscheiden, welches Verfahren bevorzugt verwendet wird.

Auch in der Testautomatisierung gibt es keine allgemeingültige Norm. Meist sind die Vorgaben der Unternehmen anzuwenden und von Software zu Software ist zu entscheiden, wie Testprozesse am besten automatisiert werden können.[4]

---

1 http://www.duden.de/rechtschreibung/Norm (Stand 23.04.2016)
2 Bucsics, Thomas et al.: Basiswissen Testautomatisierung, S3
3 Vgl. Bucsics, Thomas et al.: Basiswissen Testautomatisierung, S3
4 Vgl. Bucsics, Thomas et al.: Basiswissen Testautomatisierung, S3

2.2 Fachliche Standards und Normen

Für WMSe und WMTSe gibt es mehrere fachliche Standards und Normen. International gelten einzelne ISO-Normen und die Spezifikationen der OGC. Die Dienste des LDBV müssen zusätzlich zu diesen Standards noch den Vorgaben der AdV entsprechen.

2.2.1 International Organization for Standardization

Die ISO ist eine Vereinigung von Standardisierungsgremien aus 162 Ländern[5]. Der Name ISO steht nicht für eine Abkürzung, sondern leitet sich vom griechischen Wort „ísos“ ab[6], was so viel wie „gleich“ bedeutet.

Die ISO-Standards helfen, die Normung weltweit zu fördern und zu koordinieren, um dadurch den internationalen Handel effizienter zu machen. Sie erarbeiten auch die ISO-Normen, die von den einzelnen Mitgliedsländern unverändert übernommen werden sollen. In der Bundesrepublik Deutschland werden solche übernommenen Standards DIN EN ISO genannt, wobei DIN für das Deutsche Institut für Normung und EN für europäische Normung stehen. Seit 1993 arbeitet das TC211, ein technisches Komitee, im Bereich der Geoinformatik an den Normungen. Besonders wichtig dabei ist die Norm-Serie 191xx mit momentan 63[7] verschiedenen Standards, wobei viele dieser Normen in Zusammenarbeit mit der OGC entstanden sind.

Nach Aussagen von Frau Dr. Feichtner, Mitarbeiterin in der Geschäftsstelle GDI-BY, sind folgende ISO-Standards für die GDI-BY relevant. Diese werden nachstehend erläutert.

---

5 http://www.iso.org/iso/home/about/iso_members.htm (Stand: 28.06.2016)
6 http://www.iso.org/iso/home/about.htm (Stand: 28.06.2016)
7 http://www.iso.org/iso/home/store/catalogue_tc/catalogue_tc_browse.htm?commid=54904&published=on (Stand 26.04.2016)

In der ISO 19115 „Metadata“ werden die Standards von Metadaten für die Geodaten definiert und in der ISO 19119 „Geographic Information - Services“ für die Services konkretisiert. Die aktuelle in Deutschland verwendete Version der ISO 19115 ist aus dem Jahre 2003. Eine deutschlandweite Anpassung auf eine aktuellere[8] ISO-Version bringt einen enormen Aufwand mit sich, der wirtschaftlich nicht vertretbar ist[9].

Die ISO 19139 „Metadata - Implementation specifications“ ist eine technische Spezifikation. Sie definiert auf Grundlage der ISO 19115 ein UML Implementierungsmodell und ein XML-Schema.[10]

Für WMSe ist die ISO 19128 „Web Map service interface“ sehr wichtig. In ihr werden die technischen Grundlagen des Dienstes definiert und anhand einzelner „best practice“-Beispiele erklärt. Inhaltlich entspricht sie sehr der OGC-Spezifikation Version 1.3.0 für einen WMS[11] (siehe Kapitel 3.1).

2.2.2 Open Geospatial Consortium

Das OGC ist ein freiwilliger, weltweiter Zusammenschluss von Verwaltungen, Industrie und Wissenschaft zur Entwicklung offener Standards im Geoinformationswesen[12]. Es wurde 1994 in den Vereinigten Staaten gegründet[13].

Mittlerweile gehört das OGC zu einer der wichtigsten Institutionen, wenn es um die Schaffung und Definition von Schnittstellen geht. Gegenwärtig hat die OGC 519[14] Mitglieder. Im Gegensatz zur International Organization for Standardization gehören zu den Mitgliedern der OGC auch Unternehmen und Behörden, so zum Beispiel die European Space Agency (ESA), das Bundesamt für Kartographie und Geodäsie (BKG), AutoCAD, Google und IBM.

---

8 ISO 19115-1:2014, ISO 19115-2:2009
9 Nach Gespräch mit Frau Dr. Feichtner am 02.05.2016
10 Dressmann et al.: Übersicht der ISO Standards zu Geographischen Informationen / Geomatik, S.20
11 http://www.opengeospatial.org/standards/wms
12 https://geoportal.brandenburg.de/servicebereich/glossar/#gl_OGC
13 Vgl. http://www.opengeospatial.org/ogc/historylong
14 http://www.opengeospatial.org/ogc/members (Stand 25.04.2016)

Von den Mitgliedern werden die „OGC® Standards“ entwickelt. Die verfügbaren Spezifikationen für die Implementierung von Diensten sind weltweit offen und können kostenlos genutzt werden[15]. Die Spezifikationen beschreiben ganz detailliert die Schnittstellen und die Encodings[16] und bauen praktisch auf den ISO-Normen auf.[17] Hierbei sei zu erwähnen, dass die Standards der OGC eher als Richtlinien anzusehen sind. Ganz im Gegensatz zu den ISO-Normen, welche umgesetzt werden müssen.

Die OGC-Standards, welche für diese Arbeit verwendet wurden, sind die für den Web Map Service[18] und den Web Map Tile Service[19].

2.2.3 Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland

In der Bundesrepublik Deutschland obliegt den Ländern die Verantwortung für das amtliche Landes- und Liegenschaftsvermessungswesen[20]. Deshalb werden besondere Herausforderungen an die Zusammenarbeit von Bund und Ländern gestellt. Die Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland, kurz AdV, wurde gegründet, um fachliche Angelegenheiten von grundsätzlicher und überregionaler Bedeutung einheitlich für alle Bundesländer zu definieren. Zu den Aufgaben der AdV gehören unter anderem die Erarbeitung von Empfehlungen und verbindlichen Regelungen für ein einheitliches Vorgehen bei der Schaffung, Erhaltung und Weiterentwicklung der Geobasisdaten.[21]

Die AdV stellt auch die Codes der European Petroleum Survey Group Geodesy (EPSG-Codes) zur Verfügung[22]. Diese EPSG-Codes bestehen aus eindeutigen 4- bis

[15] Vgl. http://www.opengeospatial.org/standards
[16] Vgl. http://www.opengeospatial.org/standards (01.06.2016)
[17] Nach Gespräch mit Frau Jud am 01.06.2016
[18] http://www.opengeospatial.org/standards/wms
[19] http://www.opengeospatial.org/standards/wmts
[20] GG Art.30, Art. 70, Art. 73
[21] Vgl. http://www.adv-online.de/Wir-ueber-uns/Aufgaben/ (18.04.2016)
[22] http://www.adv-online.de/AdV-Produkte/ (23.05.2016)

5-stelligen Schlüsselnummern, die für jeweils ein bestimmtes Koordinatenreferenzsystem oder einen anderen geodätischen Datensatz stehen. Die Gauß-Krüger-Zone 4 des Deutschen Hauptdreiecksnetzes hat zum Beispiel den Code 31468[23].

Um bundesweit einheitliche Produkte anzubieten, unterliegen die Dienste der AdV gewissen Standards. Diese Standards sind in den AdV-Profilen und den AdV-Produkt-Spezifikationen zusammengefasst.

Die AdV-Profile und Produkt-Spezifikationen bilden die technische Grundlage der Bereitstellung von Geobasisdaten über Geodatendienste[24] und dienen großteils dafür, den technischen Rahmen für die Dienste festzulegen.

[23] http://spatialreference.org/ref/epsg/31468/ (23.05.2016)
[24] Vgl. http://www.adv-online.de/AdV-Produkte/Standards-und-Produktblaetter/AdV-Profile/ (18.04.2016)

3 OGC-Dienste

OGC-Dienste sind WebServices, die durch das Open Geospatial Consortiums (OGC) spezifiziert wurden und dort als beschriebene Schnittstellen-Standards hinterlegt sind[25]. Diese können in drei verschiedene Arten unterteilt werden: Darstellungsdienste, Downloaddienste und Suchdienste[26].

Bei einem WMS oder einem WMTS werden Geodaten angefordert und in Form von Rasterbildern am Bildschirm dargestellt. Deshalb gehören sie in die erstgenannte Kategorie.

Zu einem Downloaddienst zählt zum Beispiel ein Web Feature Service (WFS). Dieser liefert unter anderem Features (Vektorobjekte) im GML-Format[27].

Catalogue Services for Web (CSW), ein Suchdienst der OGC, dienen der Suche nach Geodaten und Anwendungen auf der Grundlage von Metadaten[28].

Wie bereits erwähnt, legt das OGC Spezifikationen für die einzelnen Dienste fest. In diesen Spezifikationen werden einzelne Operationen (Anfragemöglichkeiten) definiert, die von den Diensten unterstützt werden können und müssen. Die Anfrage an diese Operationen geschieht über das Hypertext Transfer Protocol (HTTP)[29].

[25] http://www.geodaten.niedersachsen.de/portal/live.php?navigation_id=27222&article_id=108069&_psmand=28
[26] Vgl. Koordinierungsstelle GDI-NI (2014): Hinweise zum Verwenden von OGC-Webservices, S. 2
[27] Vgl. Koordinierungsstelle GDI-NI (2014): Hinweise zum Verwenden von OGC-Webservices, S. 24
[28] http://www.opengeospatial.org/standards/cat (Stand 20.5.2016)
[29] Vgl. Koordinierungsstelle GDI-NI (2014): Hinweise zum Verwenden von OGC-Webservices, S. 6

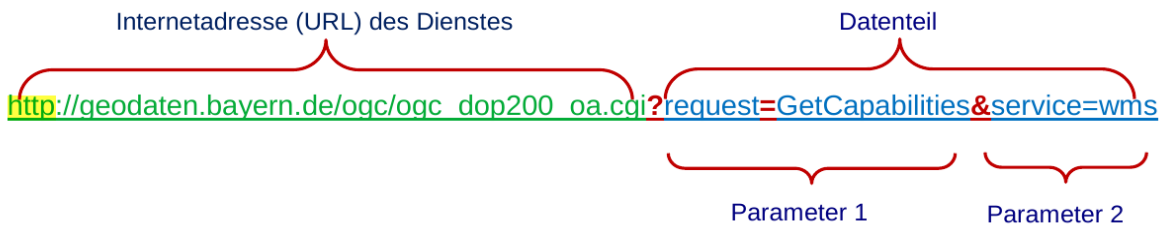

Abbildung 1: Aufbau der HTTP-Anfrage an einen WMS

Quelle: Nutzung von Geodatendiensten: Leitfaden, S.7

Für diese Anfrage wird die „GET-Methode“ als Übertragungsmethode gewählt. Das bedeutet, dass die jeweiligen Parameter nach dem Trennzeichen „?“ an die URL angehängt werden.[30]

Nach dem Fragezeichen werden die vorgegebenen Parameter der OGC zur Differenzierung der Anfrage benutzt.

3.1 Web Map Service

Ein Web Map Service (WMS) ist, ein webbasierter Darstellungsdienst, der Geodaten mit der Hilfe von georeferenzierten Bildern darstellt.[31] Die drei Operationen für einen WMS der Version 1.1.1 sind, gemäß OGC-Spezifikation, GetCapabilities, GetMap und GetFeatureInfo[32]. Die beiden ersteren Operationen müssen zwingend bedient werden. Die GetFeatureInfo jedoch ist optional[33].

Nachstehend werden nur die zwingenden Parameter für die oben genannten Operationen erläutert. Details über die Benutzung der optionalen Parameter können in den jeweiligen Spezifikationen nachgelesen werden.

[30] Vgl. OpenGIS® Web Map Service Implementation Specification, Version 1.1.1 (2002) S. 12
[31] https://www.lvermgeo.sachsen-anhalt.de/de/geoservice/wms/main.htm (19.04.2016)
[32] Vgl. OpenGIS® Web Map Service Implementation Specification, Version 1.1.1 (2002) S.21
[33] Vgl. OpenGIS® Web Map Service Implementation Specification, Version 1.1.1 (2002) S.1

3.1.1 GetCapabilities

Bei einer GetCapabilities-Abfrage werden die Eigenschaften des Dienstes abgefragt. Für die Anfrage über HTTP stehen dem Benutzer vier Parameter zur Verfügung, wobei zwei davon zwingend und zwei optional sind.

Request Parameter	Required/ Optional	Description
VERSION = version	O	Request version
SERVICE = WMS	R	Service type
REQUEST = GetCapabilities	R	Request name
UPDATESEQUENCE = string	O	Sequence number or string for cache control

Tabelle 1: Parameter bei einem GetCapabilities-Aufruf für einen WMS[34]

Der „Request“, also die Art der Anfrage, ist immer zwingend. Der zweite erforderliche Parameter ist „Service“, mit dem der Service Typ definiert wird. Für eine GetCapabilities-Anfrage des DOP80-WMS wird der Aufruf wie folgt zusammengestellt:

> *http://www.geodaten.bayern.de/ogc/ogc_dop80_oa.cgi?SERVICE=WMS&REQUEST=GetCapabilities*

Die Antwort auf diesen Request ist ein Textdokument im XML-Format. In diesem stehen sowohl allgemeine Informationen, wie zum Beispiel der Dienstanbieter oder die Gebührenordnung, als auch die technischen Daten des Dienstes, etwa die räumliche Verfügbarkeit der Layer, die abrufbaren Dateiformate und Koordinatensysteme oder die URL zum Aufruf der Legende.

[34] OpenGIS® Web Map Service Implementation Specification, Version 1.1.1, S.21

Abbildung 2: Auszug des Response der GetCapabilities Abfrage vom DOP80 WMS als XML-Dokument

3.1.2 GetMap

Bei einer GetMap-Anfrage fordert der Anwender, Geodaten in Form einer Karte, also als georeferenziertes Rasterbild, an[35]. Auch hier kann die Abfrage über den Browser getätigt werden, um das Ergebnis direkt sichtbar zu machen. Jedoch besteht der Request einer GetMap-Anfrage aus mehreren Parametern als bei GetCapabilities.

Die benötigten Werte für die Parameter der GetMap-Anfrage können aus dem Capabilties-Dokument entnommen werden.

Request Parameter	**Required/ Optional**	**Description**
VERSION = version	R	Request version
REQUEST = GetFeatureinfo	R	Request name
LAYERS=layer_list	R	Comma-separated list of one or more map layers. Optional if SLD parameter is present.
STYLES=style_list	R	Comma-separated list of one rendering style per requested layer. Optional if SLD parameter is present.

[35] Vgl. Koordinierungsstelle GDI-NI (2014): Hinweise zum Verwenden von OGC-Webservices, S. 10

Request Parameter	Required/ Optional	Description
SRS=namespace:identifier	R	Spatial Reference System.
BBOX=minx,miny,maxx,maxy	R	Bounding box corners (lower left, upper right) in SRS units.
WIDTH=output_width	R	Width in pixels of map picture.
HEIGHT=output_height	R	Height in pixels of map picture.
FORMAT=output_format	R	Output format of map.
TRANSPARENT=TRUE|FALSE	O	Background transparency of map (default=FALSE).
BGCOLOR=color_value	O	Hexadecimal red-green-blue color value for the background color (default=0xFFFFFF).
EXCEPTIONS=exception_format	O	The format in which exceptions are to be reported by the WMS (default=SE_XML).
TIME=time	O	Time value of layer desired.
ELEVATION=elevation	O	Elevation of layer desired.
Other sample dimension(s)	O	Value of other dimensions as appropriate.
Vendor-specific parameters	O	Optional experimental parameters.

Tabelle 2: Parameter bei eines GetMap-Aufrufs für einen WMS[36]

Wie schon bei der GetCapabilities-Abfrage sind die Parameter „Request“ und „Service“ zwingend, jedoch muss hier ebenfalls die „Version“ als Konstante beantwortet werden. Die Größe des angeforderten Kartenausschnitts kann über die Parameter „Width“ und „Height“ geregelt werden und die Kartengrundlage über das Wort „Layers“. Für die richtige Darstellung der Karte werden die Übergabewerte „SRS“ und „BBox“ benötigt. Mit diesen definiert man das gewünschte Koordinatensystem

[36] OpenGIS® Web Map Service Implementation Specification, Version 1.1.1, S.33

und die räumliche Ausdehnung. Durch den Style-Parameter wird es möglich, dem Benutzer einen Layer in unterschiedlichen Signaturierungen bereitzustellen.

Mit dem Parameter „Format“ wird das Ausgabeformat des Layers ausgewählt. Hierbei ist darauf zu achten, dass das gewünschte Format als MIME-Type geschrieben wird. Genaueres zu dieser Codierung von Webdokumenten wird im Kapitel 6.3.1. erläutert.

Der folgende GetMap-Aufruf des DOP80 WMS liefert ein 1000x1000 Pixel großes Bild im PNG-Format für den farbigen DOP-Layer im GK4-Koordinatenreferenzsystem.

> *http://www.geodaten.bayern.de/ogc/ogc_dop80_oa.cgi?SERVICE=WMS&VERSION=1.1.1&REQUEST=GetMap&LAYERS=by_dop80c&FORMAT=image/png&SRS=EPSG:31468&BBOX=4360000,5270000,4380000,5290000&WIDTH=1000&HEIGHT=1000&STYLES=default*

Der Styles-Parameter wird in diesem Aufruf mit dem Wert „default“ belegt. Das liegt, daran, dass bei den Diensten des LDBV nur der Standard-Wert „default“ angeboten wird. Dieser Parameter müsste bei den einzelnen Abfragen auch nicht genannt werden.

Bei einer GetMap-Anfrage ist zu beachten, dass die Pixelbox nicht überschritten wird. Dies kann passieren, wenn die angegebenen Werte bei den Parametern „Height“ und „Width“ außerhalb des zulässigen Bereichs sind. Die Größe der Pixelbox ist nicht in dem Capabilities-Dokument zu finden. Jedoch wird sie meistens in den Produktdetails oder auf der Website des Anbieters angegeben.[37] Grund dieser Beschränkung ist es, eine gewisse Leistungsvorgabe zu erfüllen und auch eine gewisse Performance zu gewährleisten. So empfiehlt die AdV zum Beispiel mindestens

[37] Nach Gespräch mit Frau Clerico, Sachgebietsleiter der Produktentwicklung, am 19.5.2016

eine Ausdehnung von 1200x1200 Pixel.[38] Mit dieser Vorgabe wird nur festgelegt, welchen Wert man nicht unterschreiten sollte. Eine Obergrenze gibt es also nicht. So haben die WMSe des LDBV eine maximale Ausdehnung von 2000x2000 Pixel, jedoch hat zum Beispiel der WMS vom Bayerischen Landesamt für Denkmalpflege auf die Daten der Denkmäler eine 2500x2500 große Pixelbox[39].

Abbildung 3: .png-Datei der GetMap Abfrage vom DOP80 WMS

3.1.3 GetFeatureInfo

Bei einer GetFeatureInfo-Abfrage werden zusätzliche Informationen von einzelnen im Rasterbild dargestellten Objekten[40], auf Englisch „features“, aufgerufen. Diese können zum Beispiel der Name des Objekts oder das Aktualisierungsdatum des Karteninhalts sein.

Der GetFeatureInfo-Operator ist optional, er muss also nicht zwingend unterstützt werden. Ob ein WMS diese Abfrage unterstützt, kann aus dem Capabilities-Dokument entnommen werden. In jedem Layer-Tag (<Layer>) steht die boolsche Variable „queryable“. Diese kann nur zwei Zustände annehmen, in diesem Fall „0“ oder „1“[41]. Der Wert „0“ kann praktisch gleichgesetzt werden mit Nein. Das bedeutet, dass dieser Layer keine GetFeatureInfo-Abfrage unterstützt. Wenn der Wert „1“ ist, ist für diesen Layer eine GetFeatureInfo Auskunft möglich.

```
<Layer queryable="1"
```

Abbildung 4: Queryable-Variable vom DOP80 WMS

[38] Vgl. AdV-Festlegung zu den INSPIRE Technical Guidance View Services version 3.1, S9
[39] Vgl. http://geoportal.bayern.de/geoportalbayern/detailpage?15&resId=224e744a-ee17-426d-969c-e3f29244cf17
[40] Koordinierungsstelle GDI-NI (2014): Hinweise zum Verwenden von OGC-Webservices, S. 12
[41] Gumm et al., Einführung in die Informatik, S. 246

Eine GetFeatureInfo-Abfrage ist sehr ähnlich zu einem GetMap-Aufruf. Durch den Parameter <map_request_copy> wird die komplette GetMap-Anfrage wiederholt.[42] Deshalb müssen die Parameter „Version“, „Request“, „Bbox“, „SRS“, „Layers“, „Width“ und „Height“ eingegeben werden.

Request Parameter	Required / Optional	Description
VERSION = version	R	Request version
REQUEST = GetFeatureinfo	R	Request name
<map_request_copy>	R	Partial copy of the Map request parameters that generated the map for which information is desired.
QUERY_LAYERS=layer_list	R	Comma-separated list of one or more layers to be queried.
INFO_FORMAT=output_format	O	Return format of feature information (MIME type).
FEATURE_COUNT=number	O	Number of features about which to return information (default=1).
X=pixel_column	R	X coordinate in pixels of feature (measured from upper left corner=0)
Y=pixel_row	R	Y coordinate in pixels of feature (measured from upper left corner=0)
EXCEPTIONS=exception_format	O	The format in which exceptions are to be reported by the WMS (default=application/vnd.ogc.se_xml).
Vendor-specific parameters	O	Optional experimental parameters.

Tabelle 3: Parameter bei eine GetFeatureInfo-Aufruf für einen WMS[43]

Zusätzlich zu den bekannten Parametern, müssen noch die Übergabewerte „X“ und „Y“ belegt werden. Diese beiden legen den X- und Y-Wert des abzufragenden Punktes fest, bezogen auf die „Width“ und „Heigth“-Parameter in der aufgerufenen

[42] Vgl. OpenGIS® Web Map Service Implementation Specification, Version 1.1.1 (2002), S.40
[43] OpenGIS® Web Map Service Implementation Specification, Version 1.1.1, S.40

Pixelbox. Wieder ähnlich zu dem GetMap-Aufruf, wird mit dem Parameter „Query-Layer“ der angeforderte Layer definiert.

Als Antwort bekommt man die Sachinformationen in dem angeforderten Format.

> *http://www.geodaten.bayern.de/ogc/ogc_dop80_oa.cgi?SERVICE=WMS&VERSION=1.1.1&REQUEST=GetFeatureInfo&LAYERS=by_dop80c&FORMAT=image/png&SRS=EPSG:31468&BBOX=4360000,5270000,4380000,5290000&WIDTH=1000&HEIGHT=1000&Styles=default&FEATURE_COUNT=20&QUERY_LAYERS=by_dop80c&INFO_FORMAT=text/html&Y=2&X=2*

Auf die Anfrage in dem obigen Beispiel erhält man, für den genannten Punkt, das genaue Datum der Befliegung und die Bildflug-Nummer im HTML-Format zurück. Wie schon bei der GetMap-Abfrage liegt wieder das GK4-Koordinateneferenzsystem zugrunde.

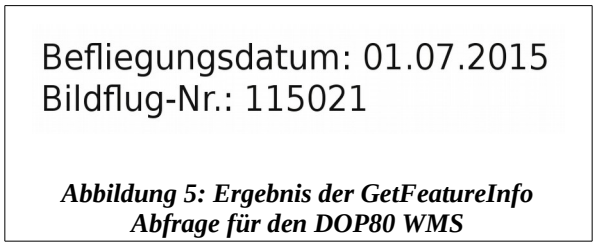

Abbildung 5: Ergebnis der GetFeatureInfo Abfrage für den DOP80 WMS

3.2 Restful Web Map Tile Service

Ein Web Map Tile Service ähnelt vom Prinzip her dem WMS, jedoch wird hier auf vorprozessierte Kacheln (256x256 Pixel[44]) von Rasterbildern zugegriffen. Dadurch sind schnelle Antwortzeiten und größere Ausschnitte möglich. Die benötigte Rechen-

[44]Vgl. OpenGIS® Web Map Tile Service Implementation Specification, S. 104

leistung ist somit geringer als bei einer WMS-Abfrage. Allerdings kann ein WMTS die Kacheln (engl. Tiles) nur in einer vom Dienstanbieter definierten Bodenauflösung, sogenannte „Zoomstufen“ bereitstellen. Da die vorgerechneten Kacheln sehr viel Speicherplatz benötigen, werden auch nur wenige Projektionen unterstützt[45].

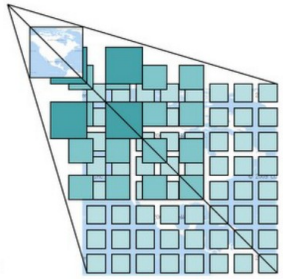

Abbildung 6: Aufbau eines WMTS

Quelle: Nutzung von Geodatendiensten, S.20

Als Abfragemethode sollte ein WMTS laut OGC-Standards RESTful oder KVP unterstützen, die SOAP-Methode ist optional[46]. Da der WMTS des LDBV nur das RESTful-Encoding unterstützt, wird auf das KVP/SOAP-Encoding an dieser Stelle nicht weiter im Detail eingegangen. Es sei nur so viel gesagt, dass bei einer Anfrage mit den prozessorientierten Schnittstellen KVP/SOAP den einzelnen Parametern Werte übergeben werden. Bei einem GetCapabilities-Request wird zum Beispiel dem Schlüssel „Service“ der Wert „WMTS“ zugewiesen. Die Übergabe erfolgt durch ein „=“, so wie bei einem WMS[47].

Bei der resourcenorientierten RESTful-Schnittstelle werden nur die einzelnen Werte angegeben. Dadurch sind die Anfragen wesentlich kürzer als mit dem KVP/SOAP-Encoding[48].

Von der OGC sind für den WMTS drei verschiedene Operationen/Ressourcen definiert. Für das KVP/SOAP-Encoding sind es GetCapabilties, GetTile und GetFeatureInfo. Beim dem REST-Encoding sind es die drei Ressourcen ServiceMetadata, Tile und FeatureInfo[49]. Im folgenden Text werden nur die Ressourcen erklärt. Details über die Operations können in den jeweiligen Spezifikationen der OGC nachgelesen werden.

[45] Vgl. Weichand, S. 1
[46] Vgl. OpenGIS® Web Map Tile Service Implementation Specification, S. 68 ff.
[47] Vgl. Weichand, S.3
[48] Vgl. Weichand, S.3
[49] Vgl. Weichand, S.2

3.2.1 ServiceMetadata

Die ServiceMetadata-Anfrage ist ähnlich zur GetCapabilities-Anfrage eines WMS. Die Eigenschaften des Dienstes werden in einer XML-Datei zurückgeliefert und beschrieben.

Die URL für eine ServiceMetadata-Anfrage kann jede Form haben, von der OGC wird jedoch die folgende empfohlen.[50]

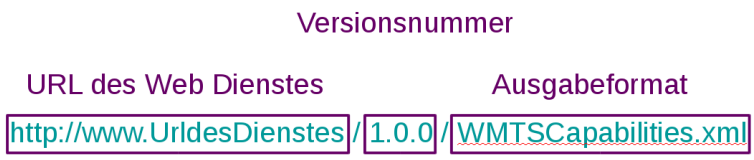

Abbildung 7: Http-Anfrage mit der RESTful-Schnittstelle

Quelle: eigene Abbildung

Wie in Abbildung 7 schon erklärt, wird bei der ServiceMetadata-Anfrage die URL des Dienstes angegeben und danach die Version. Der WMTS ist momentan nur in der Version 1.0.0 verfügbar. Anschließend folgt das Ausgabeformat, hier .xml genannt.
Für den BayernAtlas-WMTS sieht die URL wie folgt aus:

> *http://www.geodaten.bayern.de/wmts/{ServiceKey}/1.0.0/WMTSCapabilities.xml*

[50] Vgl. OpenGIS® Web Map Tile Service Implementation Specification, S.61

3.2.2 Tile

Bei einem WMTS bekommt man bei einer Tile-Anfrage eine Kartenkachel in einer festen Größe als Antwort zurückgeliefert. Es ist nicht möglich, wie zum Beispiel beim WMS, Parameter wie die Größe der Kachel anzugeben.

Request Parameter	Mandatory	Description
style	M	Style identifier
./Dimension/ ows:Identifier	M	Dimension value
TileMatrixSet	M	Tile matrix set identifier
TileMatrix	M	Tile matrix identifier
TileRow	M	Row index of tile matrix
TileCol	M	Col index of tile matrix

Tabelle 4: Parameter bei einem Tile-Aufruf für einen WMTS[51]

Wie schon beim WMS ist auch beim WMTS des LDBV der Parameter „style“ auf „default“ gesetzt. Somit kann er bei den einzelnen Abfragen weggelassen werden.

Die Parameter „Identifier“ und „TileMatrixSet“ legen den Layer und den jeweiligen Kachelsatz fest. Die „TileMatrix“ definiert die Zoomstufe und die „TileCol“ und „TileRow“-Werte die Nummer der Kachel entsprechend der Achsen der Abszisse und Ordinate.

Die Tile-Aufrufe sind immer im Capabilities-Dokument definiert. Für den BayernAtlas-WMTS wird der Aufruf wie folgt aufgebaut.

> *http://geodaten1.bayernwolke.de/wmts/{ServiceKey}/{Indentifier}/{TileMatrixSet}/{TileMatrix}/{TileCol}/{TileRow}*

> *http://geodaten1.bayernwolke.de/wmts/{ServiceKey}/by_amtl_karte/bvv_gk4/4/8/8*

[51] OpenGIS® Web Map Tile Service Implementation Specification, Version 1.0.0, S.63, Ausschnitt Table 32

Die Tile-Anfrage mit dem WMTS des LDBV sieht etwas anders aus als zum Beispiel die des BKG[52]. Am Anfang steht nicht die bekannte URL der ServiceMetadata-Anfrage, sondern der Name des Servers, auf dem die Daten liegen.

Die Angabe des Formates fehlt auch. Dies kommt daher, weil die einzelnen Layer nur ein Format unterstützen und deswegen keine weitere Konkretisierung von Nöten ist.

Abbildung 8: *.png-Datei der Tile Abfrage des BayernAtlas-WMTS

Als Ergebnis auf diese Anfrage bekommt man ein .png- Bild des Layers „Amtliche Karte“ zurück (Abbildung 8). Das Koordinatenreferenzsystem ist GK4 und die Zoomstufe ist 4. Dies entspricht ca. einem Maßstab von 1:900.000[53].

3.2.3 FeatureInfo

Diese Anfrage ist im Prinzip die Gleiche, wie die GetFeatureInfo-Anfrage bei einem WMS. Auch hier bekommt man Sachinformationen zu einem bestimmten Kartenausschnitt zurück.

Request Parameter	Mandatory	Description
style	M	Style identifier
./Dimension/ ows:Identifier	M	Dimension value
TileMatrixSet	M	Tile matrix set identifier
TileMatrix	M	Tile matrix identifier
TileRow	M	Row index of tile matrix
TileCol	M	Col index of tile matrix

[52] http://sg.geodatenzentrum.de/wmts_webatlasde.light/tile/1.0.0/webatlasde.light/webatlasde.light/DE_EPSG_25832_LIGHT/10/500/500.png

[53] Vgl. Anhang H

Request Parameter	Mandatory	Description
„J“	M	Pixel row index in a tile
„I“	M	Pixel column index in a tile

Tabelle 5: Parameter bei eine FeatureInfo-Aufruf für einen WMTS[54]

Zusätzlich zu den Parametern des Tile-Aufrufs, muss hier noch die Position des Pixels der Kachel definiert werden, an der die Sachinformation abgefragt werden soll. Dies geschieht mittels der Übergabewerte „J“ und „I“.

Der BayernAtlas-WMTS unterstützt den FeatureInfo-Request nicht. Deswegen kann hier nur das „Grundgerüst“ dargestellt werden.

> *http://URLdesDienstes/{style}/{Indentifier}/{TileMatrixSet}/{TileMatrix}/{TileCol}/{TileRow}/{J}/{I}.xml*

Von Seiten der OGC gibt es keine spezielle Vorgabe für die Reihenfolge der Variablen, es wird jedoch die obige empfohlen[55].

Als Antwort bekommt man, wie man schon am Ende der URL sehen kann, eine XML-Datei zurück.

[54] OpenGIS® Web Map Tile Service Implementation Specification, Version 1.0.0 , S. 67, Ausschnitt Table 34
[55] OpenGIS® Web Map Tile Service Implementation Specification, Version 1.0.0 , S. 67

4 Grundlagen der Testautomatisierung

4.1 Was bedeutet Testen?

Wie schon das Anfangszitat gut erklärt, bedeutet Testen nicht, die Abwesenheit von Fehlern zu beweisen. Nur weil bei einem Softwaretest alle aufgedeckten Fehler behoben worden sind, heißt das nicht, dass die Software zu 100% fehlerfrei ist. Testen soll vielmehr dazu dienen, das korrekte Verhalten einer Software aufzuzeigen. Dies kann am besten geprüft werden, indem einzelne Testfälle definiert werden und so die wichtigsten Qualitätsmerkmale getestet werden. Dazu muss dem Tester jedoch klar sein, was die Software eigentlich können soll. Meistens werden bei einem Test nur die „gängigsten“ Kombinationen getestet.

Bei einer Webapplikation sollte zum Beispiel getestet werden, ob man diese mit den verbreitetsten Internetbrowsern aufrufen kann. Hierbei ist es jedoch unausweichlich die zu testenden Versionen einzuschränken, denn es ist nicht möglich alle verfügbaren Systemkonstellationen abzudecken. Des Weiteren müssen auch die aktuellen Versionen berücksichtigt werden. Zum Beispiel ist es nicht sehr produktiv, wenn die Applikation mit dem Internet Explorer 8 (2008) funktioniert, hingegen bei Microsoft Edge (2015), der aktueller und verbreiteter ist, gravierende Fehler auftreten.

Dieses Beispiel zeigt auf, dass die Aussage „eine Software sei komplett getestet worden“, nicht zutreffend ist. Es gibt einfach zu viele Möglichkeiten und Konstellationen, als dass man alle Eventualitäten testen könnte. Richtig hingegen ist die Aussage, dass die Software die im Vorfeld festgelegten Qualitätsmerkmale erfüllt und dass eine einwandfreie Nutzung, zum Beispiel mit den verbreitetsten Betriebssystemen, gewährleistet ist.

Zudem sollte ein Test immer ausreichend dokumentiert sein, seien es die einzelnen Testfälle oder das Feedback der Tester.

4.2 Die Psychologie des Testens

Der Entwickler und der Tester sollten immer zwei verschiedene Personen sein, nur dann ist eine gewisse Objektivität gewahrt. Ein Tester testet ohne Rücksicht auf den Entwickler. Da der Entwickler das Programm ja selbst geschrieben hat, kann er es nicht objektiv beurteilen[56]. Es wird ihm schwerfallen, seine Fehler als wirkliche Fehler anzuerkennen oder er wird schlichtweg seine Fehler übersehen. Außerdem kann es auch sein, dass er die Aufgabenstellung falsch verstanden hat, so dass ihm bei einem Test gar kein Fehler auffallen würde[57].

Zudem haben die beiden Personengruppen auch eine unterschiedliche Betrachtungsweise auf das Testverfahren[58]. Die Tester versuchen bei einem Test einen Fehler der Software nachzuweisen, ein Entwickler jedoch versucht beim testen, das korrekte Verhalten zu beweisen.

Für ein „harmonisches“ Testverfahren dürfen sich die einzelnen Beteiligten, also die Entwickler und die Tester, nicht als Gegner ansehen. Denn wenn der Entwickler in den Testern nur Erbsenzähler sieht oder die Tester sich darüber ärgern, was der Entwickler denn „da schon wieder programmiert hat“, kann dies zu unnötigen Komplikationen führen. Um dem vorzubeugen, sollten beide friedlich koexistieren und sich gegenseitig auch respektieren. Der Tester sollte seine Kritik konstruktiv gestalten[59] und der Entwickler darf nicht vergessen, dass sein Programm auf dem Prüfstand steht und nicht er selbst. Dies bedeutet auch, dass er die Kritik nicht persönlich nehmen darf.

Hieraus folgt ein weiterer Vorteil der automatisierten Tests - Sie sind emotionslos!
Ein automatisierter Test zeigt nur den aktuellen Ist-Stand an, und dies komplett wertfrei. Wenn dem zuständigen Entwickler für den Web-Dienst das Ergebnis des Tests

[56] Vgl. Bucsics, Thomas et al.: Basiswissen Testautomatisierung, S. 35
[57] Vgl. Bucsics, Thomas et al.: Basiswissen Testautomatisierung, S. 35
[58] Vgl. Bucsics, Thomas et al.: Basiswissen Testautomatisierung, S. 34
[59] Vgl. Bucsics, Thomas et al.: Basiswissen Testautomatisierung, S. 36

mitgeteilt worden ist, erfährt er nur wo noch eine Diskrepanz zwischen dem Soll- und Ist-Stand ist. Das sind einfache Fakten ohne jegliche Wertung.

4.3 Was ist Testautomatisierung?

Bei der Testautomatisierung werden einzelne Tests, wie es der Name schon sagt, automatisiert, also mittels einer Software getestet. Wobei man sich nicht der Illusion hingeben darf, dass dadurch alles besser und leichter wird oder dass damit nie wieder etwas manuell getestet werden muss. Denn dies stimmt leider nicht, aber hierauf wird später genauer eingegangen.

Bevor eine Software automatisiert getestet werden soll, ist vorher klarzustellen, was genau getestet werden soll. Prinzipiell kann alles automatisiert getestet werden, vom Modultest bis zum Integrationstest. Ist diese Frage beantwortet, dann ist zu klären, welche Software zum Einsatz kommen soll oder ob gegebenenfalls eine Eigenentwicklung die bessere Lösung darstellt. Der Markt wird momentan überschwemmt von derartiger Software, sowohl kostenpflichtige wie auch open-source-Lösungen[60].

Es wird deutlich, dass im Vorfeld durchaus vieles erst geklärt werden muss, bevor mit Automatisierungen begonnen werden kann[61]. Einen Testprozess zu automatisieren ist nichts, was schnell mal umgesetzt werden kann. Vor allem können trotz intensiver Vorbereitung und Klärung der wichtigsten Fragen Rückschläge oder grundlegende konzeptionelle Überarbeitungen vorkommen.

4.4 Grenzen der Automatisierung

Eine Testautomatisierung dient nicht dazu „normale“, also menschliche Tester, zu ersetzten. Häufig wird das falsch verstanden und versucht alles zu automatisieren und

[60] https://www.testing-board.com/testautomatisierung-tools/, besucht am 20.05.2016
[61] Vgl. Bucsics, Thomas et al.: Basiswissen Testautomatisierung, S. 11

damit die Tester „wegzurationalisieren“. Dies ist ein schwerwiegender Fehler, denn Testen ist etwas Kreatives und Intuitives[62], eine Maschine jedoch ist nicht in der Lage, eigenständig zu denken bzw. zu handeln. Sie testet nur das, was ihr vorgegeben wird. In den vorliegenden Fällen dieser Arbeit wird verglichen, ob die Angaben, die vom Auftraggeber erstellt wurden, mit den Daten im Capabilities-Dokument übereinstimmen. Wenn dabei zum Beispiel ein Fehler unterläuft und die Soll-Werte falsch festlegt wurden, fällt das der Maschine nicht auf. Außerdem kann man mit einem automatisierten Test nur Dinge prüfen, die einem selbst einfallen oder die durch konkrete Normen überprüft werden müssen. Bei einer Software können jedoch auch unerwartete Fehler auftreten, die einem im Vorfeld nicht mal ansatzweise in den Sinn gekommen wären.

Des Weiteren ist zu klären, dass es Entwicklungen gibt, bei denen sich eine Automatisierung nicht rentiert. Zum Beispiel wird bei den automatisierten Tests am LDBV der Inhalt des GetLegend- Aufrufs eines WMS nach wie vor von den Testern überprüft. Tester können mit einem Blick beurteilen, ob die Legende korrekt wiedergegeben wird. Bei den automatisierten Tests jedoch müsste wahrscheinlich erst ein Histogramm mit den Soll-Werten vorgeben werden und dieses dann mit dem Histogramm, von der GetLegend-Abfrage aus dem Capabilities-Dokuments, vergleichen werden. Selbst bei diesem Verfahren, hat man am Ende trotzdem keine Sicherheit, dass der Aufruf dann korrekt ist.

Der wichtigste Grund, der jedoch gegen eine vollständige Automatisierung spricht, ist dass die Software bzw. die Dienste schlussendlich von Menschen verwendet werden. Also können nur Menschen eine Aussage über Nutzerfreundlichkeit und Bedienbarkeit treffen. Und auch nur menschliche Tester können versuchen, die Software mittels Optimierungsvorschlägen zu verbessern.

[62] Vgl. Spillner, Andreas et al.: Basiswissen Softwaretest S. 35

5 Entwicklung von automatisierten Tests

5.1 Realisierung und Durchführung der Tests

Im Zuge der Entwicklung des BayernAtlas-WMTS kam das Thema Testautomatisierung zum ersten Mal auf. Den Mitarbeitern des Sachgebiets QM-IuK fiel es schwer, Testvorgaben zu entwickeln, da die meisten Tester in der BVV im Umgang mit WMTSen noch unerfahren waren. Der Vorschlag der Automatisierung kam von Seiten der Entwickler. Ihre Idee war es auch, die Automatisierung mittels der OWSLib[63], einer Open-Source Bibliothek für Python, umzusetzen.

Die verwendete Programmiersprache Python arbeitet mit effizienten abstrakten Datenstrukturen und einem einfachen, aber effektiven Ansatz zur objektorientierten Programmierung[64]. Es ist eine dynamische Programmiersprache und arbeitet somit ohne Deklaration von Typen[65]. Dadurch werden zum Beispiel weniger Codezeilen benötigt.

Zu Beginn musste das vorhandene Testkonzept angesehen und herausgefiltert werden, was generell automatisiert werden könnte. Die einzelnen Punkte wurden dann in eine Liste aufgenommen. Anhand dieser Liste wurde dann ausgewertet, was sich wie schnell umsetzten lässt und welche Punkte die größte Dringlichkeit haben.

Daraufhin wurden die Programmcodes für WMS und WMTS in der OWSLib untersucht und geprüft, ob man mit dem Originalcode arbeiten kann, oder ob etwas abgeändert werden müsste. Die Codes von der OWSLib dienen primär dazu, das Capabilities-Dokument eines Dienstes auszulesen und mit den vorprogrammierten Parametern anschließend die einzelnen Informationen abzufragen. Dies können zum Beispiel die vorhandenen Layer sein oder auch der Name des Dienstes.

[63] https://geopython.github.io/OWSLib/ (Stand: 21.06.2016)
[64] Vgl. https://py-tutorial-de.readthedocs.io/de/python-3.3/ (Stand: 22.05.2016)
[65] http://www.lesscode.de/initiative/dynamische-programmiersprachen/ (Stand: 22.05.2016)

Das Prinzip der Tests ist ein einfacher Soll-Ist-Vergleich. Für fast jeden Dienst des LDBV gibt es eine Produktspezifikation, die vom Auftraggeber, der Vertriebsabteilung verfasst wurde. In den Tests werden die Werte aus den Spezifikationen als Soll-Werte vorgegeben. Die vorprogrammierten Parameter vergleichen diese Werte dann mit den Werten im Capabilities-Dokument.

In den von mir erstellten Testdokumenten werden nicht nur die Soll-Werte vorgegeben, sondern auch die Befehle, was genau aus dem Capabilities-Dokument ausgelesen werden soll. Wenn man die Tests startet, werden alle Befehle der Reihe nach ausgeführt und somit die beiden Werte verglichen.

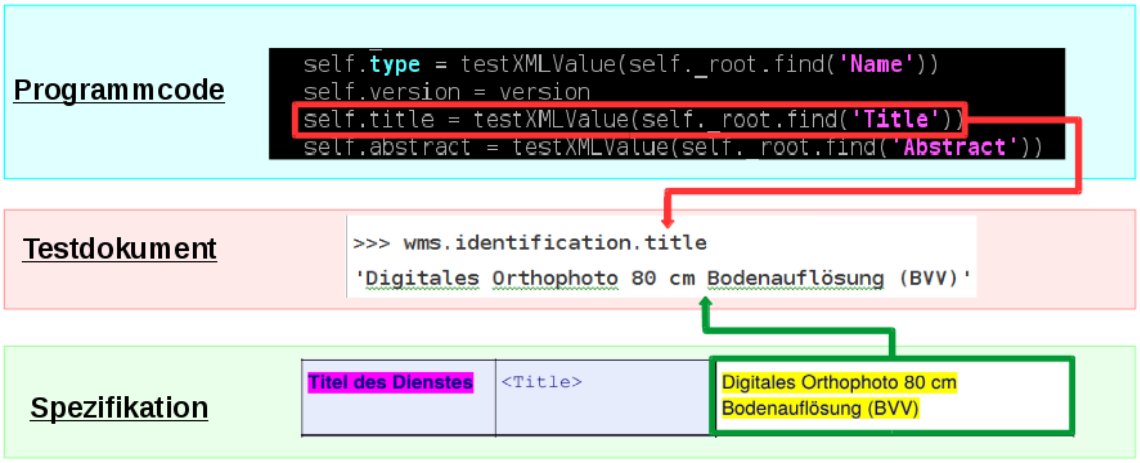

Abbildung 9: Zusammenstellung der Tests

Quelle: eigene Abbildung

Wenn ein Testdokument fertiggestellt wurde, prüfe ich es zuerst auf meinem lokalen Rechner. Dafür wird der Test über die Shell[66] gestartet und das Ergebnis ausgewertet. Wenn das Ergebnis des Tests fehlerfrei ist, also der Soll-Wert gleich dem Ist-Wert ist, wird das Dokument auf den Server hochgeladen. Dort wird es dann mittels Jenkins[67] automatisiert geprüft. Wenn der Test jedoch nicht fehlerfrei durchlief, sende ich die Auswertung an den zuständigen Entwickler weiter mit der Bitte, den Fehler zu

[66] Siehe Glossar der Fachwörter
[67] Siehe Glossar der Fachwörter

beheben. Wenn dies geschehen ist, geht der Zyklus von vorne los (siehe Abbildung 10).

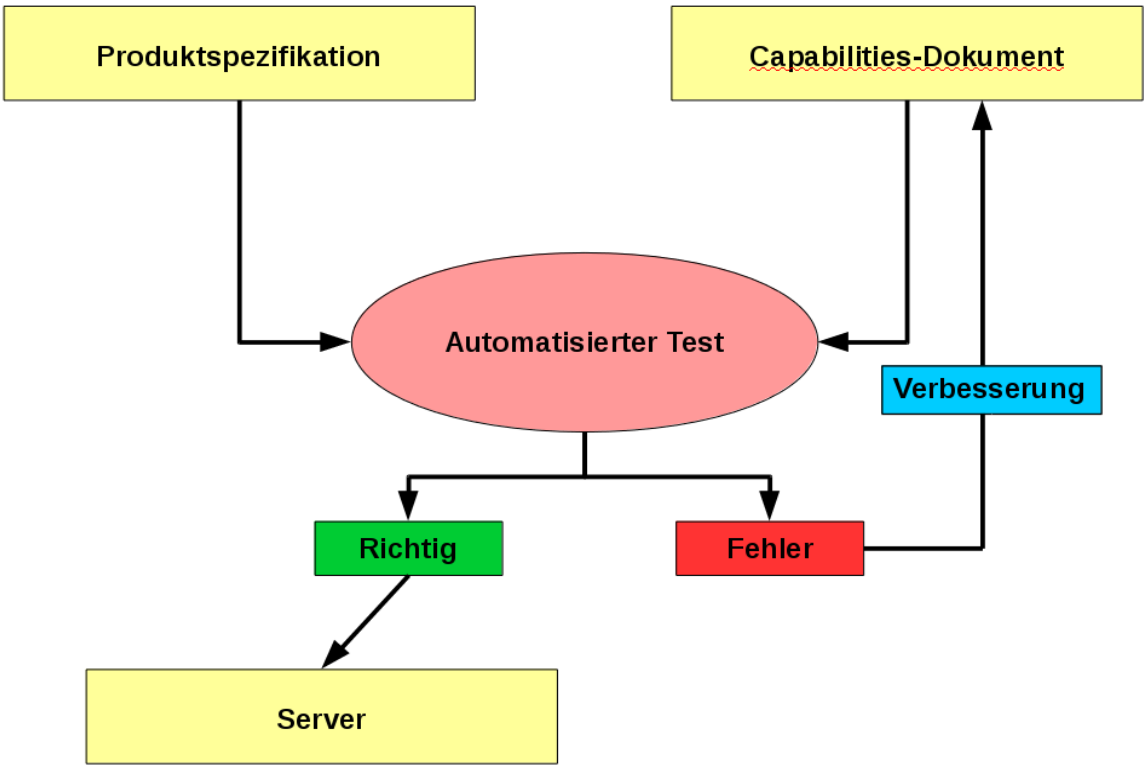

Abbildung 10: Ablaufschema der Erstellung der automatisierten Tests

Quelle: eigene Abbildung

Die Dateien, die in Jenkins vorhanden sind, werden momentan jeden Tag in der Früh um 8 Uhr automatisch überprüft. Es ist jedoch auch möglich, jederzeit manuell einen neuen „build[68]“ zu erzeugen.

Jeder dieser „builds“ generiert eine neue .html-Seite[69], in welcher die einzelnen Dienste aufgelistet sind. Anhand dieser Liste kann überprüft werden, ob der Dienst funktionstüchtig ist oder nicht. Dies muss jedoch manuell geprüft werden, da Jenkins

[68] Siehe Glossar der Fachwörter
[69] Siehe Anhang G: Jenkins Report vom 12.04.2016

die Tests nur ausführt und nicht die Ergebnisse bewertet oder Fehlerbeseitigungen anstößt. Im Gegensatz zu einem Monitoring-Programm, wird auch nicht Alarm geschlagen, wenn ein Dienst nicht funktioniert.

5.2 Aufbau des Tests

Momentan konzentriert sich der Test nur darauf, ob das Capabilities-Dokument korrekt ist. Für die Abfrage der einzelnen Operationen, wird augenblicklich nur geprüft, ob die URL, die im Capabilities-Dokument steht, stimmt, also ob zum Beispiel ein GetMap-Aufruf prinzipiell möglich wäre.

Zunächst wurde für jeden WMS und WMTS ein Testdokument erstellt, welches dann mit Jenkins überprüft wurde. Dieses Konzept hat sich jedoch schnell als falsch herausgestellt, da bei der Auswertung mittels Jenkins immer nur der erste Fehler angezeigt wird[70]. Dadurch kann es sein, dass „akzeptable“ Fehler, wie zum Beispiel ein Rechtschreibfehler, angezeigt werden und gravierende Fehler, die die Funktionen des Dienstes beeinträchtigen, übersehen werden.

Damit dies nicht passiert, wurden die bereits erstellen Testdokumente an manchen Stellen umgeschrieben und auf drei Dateien, „Prosa“, „Layer“ und „Base“, aufgeteilt.

Die Idee dahinter war, zwischen schwerwiegenden und nebensächlichen Fehlern, die auftreten können, zu unterscheiden.

In dem Testdokument „Prosa“, werden die Inhalte geprüft, die Rechtschreibfehler haben können und die zur Funktionsfähigkeit eines Dienstes nichts beitragen. Also zum Beispiel die Beschreibung des Dienstes oder die Kontaktinformationen des Kundenservices des LDBV.

[70] Vgl. Anhang G

Bei „Layer“ werden die Informationen geprüft, die die einzelnen Layer beschreiben und fehlerhaft sein können. So zum Beispiel die Parameter „Keywords“ oder „Abstracts“

Die wichtigste Datei von allen ist „Base“. Mit ihr werden nur Angaben geprüft, die maßgeblich daran beteiligt sind, dass der Dienst funktionsfähig ist. Zum Beispiel die Größe der BoundingBox oder die zu unterstützenden Koordinatenreferenzsysteme. Dadurch erfolgt die Auswertung schneller und es ist möglich auf einen Blick zu erkennen ob bei einem Dienst eine Störung vorliegt.

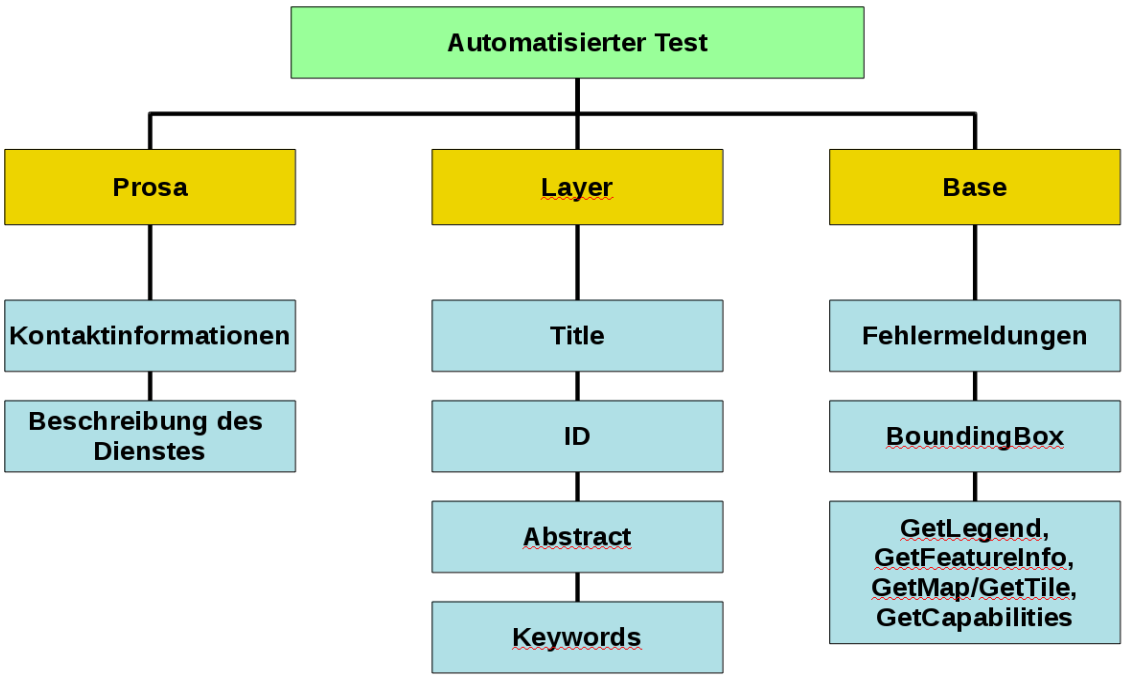

Abbildung 11: Aufbau der automatisierten Tests

Quelle: eigene Abbildung

Wenn die Tests erweitert werden, kann es aber durchaus sein, dass dieser Aufbau angepasst werden muss.

5.3 Anforderungen an den Test

Der wichtigste Aspekt für die automatisierten Tests war, die Testverfahren von den WMSen und dem WMTS zu unterstützen bzw. zu vereinfachen. Die menschlichen Tester sollten entlastet werden und sich auf Anwendungstests konzentrieren und auf Aspekte, für die eine Automatisierung zu komplex und zu aufwendig wäre. Wie zum Beispiel den Inhalt eines GetLegend-Aufrufs oder allgemeine Unregelmäßigkeiten.

Die Prüfung des GetMap- oder GetFeatureInfo-Aufrufs kann momentan noch von menschlichen Testern übernommen werden, denn dafür müssen sie nur den Dienst in einen geeigneten Client, zum Beispiel den BayernAtlas[71] einbinden und prüfen ob sie ein Bild oder durch einen Klick in die Karte eine Objektinformation erhalten. Der Test des Capabilities-Dokuments jedoch ist prädestiniert für Lesefehler. Egal ob man das Dokument im XML-Format per Eingabe der URL oder als HTML-Seite per Nutzung des Capabilities-Viewers[72] prüft, jedes Mal muss der Inhalt durchgelesen und mit den Spezifikationen verglichen werden. Da kann es sehr schnell passieren, dass ein Fehler überlesen wird.

Deswegen war dies der wichtigste Punkt, der automatisiert werden sollte. Für eine Maschine ist es ein leichtes, einzelne Werte zu vergleichen, seien es Zahlen oder Wörter.

Darüber hinaus sollte der automatisierte Test es ermöglichen, bei Störungen schneller einen Überblick zu bekommen, welche Funktionseinschränkung vorhanden ist und dadurch die Reaktionszeiten zu minimieren. Bevor die automatisierten Tests bei QM-IuK realisiert wurden, musste in einem Störfall immer erst manuell überprüft werden, welche Dienste betroffen sind. Durch die automatisierten Tests ist es jetzt möglich einen neuen „build“ zu erstellen und diesen auszuwerten.

[71] Siehe Glossar der Fachwörter
[72] Siehe Glossar der Fachwörter

Die Hoffnung war auch, dass durch die Tests die Entwicklung der Dienste erleichtert wird und nicht mehr bis zum „offiziellen“ Testverfahren gewartet werden muss um erste Fehler zu beheben. Die im Rahmen dieser Arbeit erstellten Testdokumente wurden von mir bereits während der Entwicklung erstellt. So kann immer wieder überprüft werden, ob alles den Normen entspricht oder ob etwaige Änderungen an anderen Stellen negative Auswirkungen haben. Durch dieses Verfahren, gibt es auch nicht mehr so viele „böse Überraschungen“ während des Testverfahrens, wie, dass zum Beispiel, dass bei einem WMS die GetMap-Operation nicht funktioniert. Viele derartigen Fehlern können mit den automatisierten Tests schon bei der Entwicklung entdeckt und berichtigt werden.

6 Automatisierte Tests am LDBV am Beispiel DOP80 WMS und BayernAtlas WMTS

Da im vorangegangenen Kapitel auf den Aufbau der Tests eingegangen wurde, werden hier an dieser Stelle nur einzelne Besonderheiten aufgezählt. Die kompletten Testdokumente sind im Anhang zu finden.

6.1 DOP80 WMS

6.1.1 Einlesen der Daten

Am Anfang eines jeden Testdokumentes werden einzelne Methoden/Funktionen aus den Programmbibliotheken importiert. Dies geschieht mit der Import-Anweisung.

Das hier zugrundeliegende Prinzip heißt modulare Programmierung bzw. Modularisierung. Es werden also Programme in logische Module unterteilt und diese werden später in den eigentlichen Programmcode importiert. Die OWSLib ist ein solches Modul.

```
1  Imports
2
3  >>> from __future__ import (absolute_import, division, print_function)
4  >>> from tests.utils import cast_tuple_int_list, cast_tuple_int_list_srs, resource_file
5  >>> from owslib.wms import WebMapService
6  >>>
7  >>> wms = WebMapService('http://www.geodaten.bayern.de/ogc/ogc_dop80_oa.cgi',version='1.1.1')
```

Abbildung 12: Import des DOP80-WMS

Quelle: eigene Abbildung

Nach dem Import wird mit dem Befehl „WebMapService ('...')“ das Capabililties-Dokument des Dienstes aufgerufen und ausgelesen. Dieser Aufruf geschieht mittels einer

Variablendeklaration. Einfachheitshalber wurden die Variablen mit WMS/WMTS benannt.

6.1.2 Prüfung der unterstützten Operationen

Die meisten WMSe des LDBV unterstützen die drei vorgegebenen Operationen der OGC und zusätzlich die Operation GetLegendGraphic. Diese wurde nicht in Kapitel 3.1 zusammen mit den anderen Operatoren erklärt. Das liegt daran, dass dieser Operator nur von Styled Layer Descriptors-WMSen unterstützt werden kann[73] und auch nur optional unterstützt werden muss. Genaueres kann man in dem Styled Layer Descriptor Profil für WMSe von der OGC nachlesen[74].

Die anderen Operationen aus Kapitel 3.1 - also GetMap, GetCapabilities und GetFeatureInfo - werden von fast allen WMSen des LDBV unterstützt.

```
146  >>> sorted([operation.name for operation in wms.operations])
147  ['GetCapabilities', 'GetFeatureInfo', 'GetLegendGraphic', 'GetMap']
```

Abbildung 13: Unterstützende Operationen des DOP80 WMS

Quelle: eigene Abbildung

Bei den Tests werden die Operatoren nach dem Alphabet sortiert abgefragt. Damit wird die Eingabe der Soll-Werte erleichtert. Wie schon in Kapitel 5.1 beschrieben, werden diese Werte den Produktspezifikationen entnommen.

[73] Vgl. OpenGIS® Web Map Service Implementation Specification, Version 1.1.1 (2002) S.42
[74] http://www.opengeospatial.org/standards/sld

6.1.3 Prüfung der URL und der Formate der einzelnen Operationen

Für jede Operation ist in den Spezifikationen vorgegeben, welche Formate unterstützt werden müssen. Wie auch bei der Anfrage der Operatoren (Kapitel 3) müssen diese Formate als MIME-Type (od. Content-Type) angegeben werden. Die Schreibweise stellt sich aus zwei Typen zusammen: Der Medientyp und der Subtyp. Bei einem Bildformat, zum Beispiel JPEG, schreibt man dann image/jpeg oder bei einem Textformat text/html.

```
114  GetMap
115
116  >>> list(sorted(wms.getOperationByName('GetMap').formatOptions))
117  ['image/jpeg', 'image/png']
```

Abbildung 14: Formate der GetMap-Abfrage des DOP80-WMS

Quelle: eigene Abbildung

Bei der GetCapabilities-Abfrage gibt es bei der Angabe des XML-Formates eine Besonderheit. Wie Abbildung 15 entnommen werden kann, wird hier nicht die Schreibweise wie bei den anderen MIME-Typen verwendet. Dies liegt daran, dass es für die Ausgabe der Capabilities einen eigenen Content-Type[75] von der OGC gibt[76]. Neben dem XML-Format für die Capabilities gibt es noch vordefinierte Schreibweisen für die „Service Exceptions“ und für das GML-Format der GetFeatureInfo.

[75] Siehe Glossar der Fachwörter
[76] Vgl. OpenGIS® Web Map Service Implementation Specification, Version 1.1.1 (2002) S.14

```
125  GetCapabilities
126
127  >>> list(wms.getOperationByName('GetCapabilities').formatOptions)
128  ['application/vnd.ogc.wms_xml']
```

Abbildung 15: Format der GetCapabilities-Abfrage des DOP80-WMS

Quelle: eigene Abbildung

Bei der Prüfung der URL muss hier auf die jeweilige Übertragungsmethode - „POST“ oder „GET“ - geachtet werden. Von Seiten der OGC sollte mindestens eine von beiden Methoden unterstützt werden[77].

Die WMSe des LDBV unterstützen beide Methoden, weswegen die URL auch zweimal im Capabilities-Dokument aufgeführt ist. Aufgrund dessen wird sie auch zweimal von den Tests abgefragt und geprüft.

```
119  >>> sorted(list(wms.getOperationByName('GetMap').methods[0].values()))
120  ['Get', 'http://www.geodaten.bayern.de/ogc/ogc_dop80_oa.cgi?']
121  >>> sorted(list(wms.getOperationByName('GetMap').methods[1].values()))
122  ['Post', 'http://www.geodaten.bayern.de/ogc/ogc_dop80_oa.cgi?']
```

Abbildung 16: GetMap-URL des DOP80-WMS

Quelle: eigene Abbildung

Mit dem Aufruf „list(wms.getOperationByName('GetFeatureInfo').methods)“ würden beide Methoden als Antwort zurückkommen. Der Korrektheit halber werden die beiden Methoden getrennt voneinander abgefragt. Die richtige Schreibweise wird in Abbildung 16 dargestellt.

[77] Vgl. OpenGIS® Web Map Service Implementation Specification, Version 1.1.1 (2002) S.11

6.1.4 Prüfung ob der Layer eine GetFeatureInfo unterstützt

Wie schon in Kapitel 3.1.3 erklärt, kann man bei der Prüfung, ob ein WMS die GetFeatureInfo unterstützt, einfach einen boolschen Wert vorgeben. Im Fall des DOP80-WMS wird die Zahl 1 vorgegeben.

```
37  >>> wms['by_dop80_info'].queryable
38  1
```

Abbildung 17: Prügung der GetFeatureInfo für den DOP80-WMS

Quelle: eigene Abbildung

Manche Dienste des LDBV, zum Beispiel der Schummerungs-WMS[78], unterstützen die GetFeatureInfo-Operation nicht. Bei diesen wird dann die Zahl 0 vorgegeben.

6.1.5 Prüfung der verfügbaren Layer

Die WMSe des LDBV besitzen mehrere Layer. Um zu gewährleisten, dass auch alle Layer vorhanden sind, werden gleich zu Beginn des Testdokumentes „Layer“ die korrekte Anzahl und die jeweiligen Namen geprüft. An dieser Stelle soll erwähnt werden, dass laut AdV-Profil[79] die unten aufgeführten Namen, die eigentlichen Layernamen sind. Die Titel, die auch in einem Desktop-GIS angezeigt werden, heißen Layertitel, also beispielsweise „DOP 80 (Farbe)“.

[78] https://geoportal.bayern.de/geodatenonline/seiten/wms_schumm
[79] Vgl. AdV-Festlegung zu den INSPIRE Technical Guidance View Services version 3.1, S. 12

```
12  >>> sorted(list(wms.contents))
13  ['by_copyright', 'by_dop80_info', 'by_dop80c', 'by_dop80g']
```

Abbildung 18: Verfügbare Layer des DOP80 WMS

Quelle: eigene Abbildung

Um den Inhalt der Layer zu untersuchen, werden bei dieser Abfrage die vorgegebenen Namen angegeben. Mit diesen können dann zum Beispiel der Layertitel, die „Keywords“ oder auch die „Abstracts“[80] kontrolliert werden.

Obwohl oben erwähnt wurde, dass die wichtigen Punkte, welche nicht fehlschlagen dürfen, in dem Testdokument „Base“ zu finden sind, findet die Prüfung der Layernamen im Dokument „Layer“ statt. Da hier auch alle anderen Angaben, die die Layer betreffen, in dieser Datei getestet werden, sollte diese Prüfung, meiner Meinung nach, auch in diesem Dokument stattfinden. Damit es aber nicht dazu kommen kann, dass ein Ladefehler bei den Layern übersehen wird, wird dies in dem Dokument „Layer“ als erstes geprüft.

6.2 BayernAtlas-WMTS

6.2.1 Prüfung der zu unterstützenden Kachelsätze

Um auch zu gewährleisten, dass die wenigen Kachelsätze, die von einem WMTS unterstützt werden, funktionieren, werden bei dem Test für jeden einzelnen Layer diese Angaben kontrolliert. Dazu werden wieder die Namen aus den Spezifikationen vorgegeben.

[80] Siehe Anhang B

```
144  >>> sorted(wmts['by_dop'].tilematrixsetlinks.keys())
145  ['bvv_gk4', 'smerc']
```

Abbildung 19: Vorhandene Projektionen für den Layer DOP

Quelle: eigene Abbildung

Bei dem BayernAtlas-WMTS sind dies die von Google definierte Web Mercator Projektion (smerc) und Gauß-Krüger Zone 4 (bvv_gk4). Die Angabe der Werte erfolgt wieder alphabetisch.

6.2.2 Prüfung der Zoomstufen

Ein WMTS besitzt im Gegensatz zum WMS Zoomstufen und keine Scale Hints, wie beim WMS 1.1.1 oder Maßstabszahlen, wie beim WMS 1.3.0. Die Anzahl der Zoomstufen wurden für die beiden Projektionen vom Auftraggeber des LDBV festgelegt[81].

```
31  >>> sorted(wmts.tilematrixsets['bvv_gk4'].tilematrix.keys())
32  ['0', '1', '10', '11', '12', '13', '14', '2', '3', '4', '5', '6', '7', '8', '9']
```

Abbildung 20: Anzahl der Zoomstufen des BayernAtlas-WMTS

Quelle: eigene Abbildung

Bei der Vorgabe der Zahlen ist darauf zu achten, dass diese als Strings[82] ausgelesen werden und nicht als Integer[83]. Deshalb darf man die Werte nicht in numerischer Reihenfolge angeben. Bei dieser Sortierung wird nach dem Wert der ersten Stelle sor-

[81] Siehe Anhang I
[82] Vgl. Glossar der Fachwörter
[83] Vgl. Glossar der Fachwörter

tiert. Also müssen die Strings so angegeben werden wie in Abbildung 19 gezeigt wird.

6.2.3 Prüfung der Anzahl der Kachelspalten und -reihen

Der Parameter „Matrixwidth“ gibt die Anzahl der Kachelspalten an und „Matrixheight“, die der Kachelreihen[84]. Für die 14. Zoomstufe des Kachelsatzes bvv_gk4 liegen diese Werte bei 16384.

```
72  >>> wmts.tilematrixsets['bvv_gk4'].tilematrix['14'].matrixwidth
73  16384
```

Abbildung 21: Anzahl der Kachelspalten des BayernAtlas-WMTS in der Zoomstufe 14

Quelle: eigene Abbildung

Die Anzahl der Reihen und Spalten wird bei einem WMTS mittels eines Zweiersystems berechnet. Das bedeutet, dass für die erste Zoomstufe (= Z0) der Wert bei 1 liegt, denn $2^0 = 1$. Folglich liegt der Wert für die Zoomstufe 14 bei 16384 denn, $2^{14} = 16384$. Für alle Zoomstufen werden diese errechneten Werte vorgegeben und dann kontrolliert.

6.2.4 Prüfung der Tile-Abfrage

Wie bei der GetLegend-Abfrage bei WMSen (Kapitel 4.4) rentiert es sich auch nicht, den Inhalt der Tile-Abfrage zu prüfen. Was jedoch kontrolliert werden kann und sollte, ist, ob für die angegebenen Parameter im Capabilities-Dokument eine Kachel zurückgeliefert wird.

[84] Vgl. ,S. 20

```
>>> tile = wmts.gettile(layer='by_amtl_karte', tilematrixset='bvv_gk4', tilematrix='4', row='8', column='8',
 format="image/png")
>>> out = open(scratch_file('wmts_amtl_karte.png'), 'wb')
>>> bytes_written = out.write(tile.read())
>>> out.close()
```

Abbildung 22: Prüfung der Tile Anfrage

Quelle: eigene Abbildung

Dafür werden in dem Test einfach die einzelnen Parameter, die für eine Tile-Abfrage benötigt werden, aufgelistet. Mittels des Programmcodes wird dann ein Bild erstellt und temporär gespeichert. Ist der Wert des Bildes > 0 KB, dann war die Abfrage erfolgreich und es ist kein leeres Bild zurückgekommen.

Die Richtigkeit des Inhalts des Bildes kann nur überprüft werden, wenn das Bild geöffnet und mit einem Referenzbild verglichen wird. Jedoch wird der Inhalt der Kacheln eher kontrolliert, indem man den WMTS in ein Desktop-GIS einbindet und so visuell prüft, ob der Dienst korrekt läuft.

7 Ausblick

7.1 Test der GetMap und GetFeatureInfo

Wie bereits erwähnt, ist der Test momentan so ausgelegt, nur die Informationen, die im Capabilities-Dokument stehen, zu überprüfen. Momentan arbeite ich daran, auch die GetMap- und GetFeatureInfo-Abfrage automatisiert zu testen. Das bedeutet, dass für jeden WMS GetMap- und GetFeatureInfo-Abfragen für alle möglichen Parameter und Konstellationen automatisch abgefragt werden. Dadurch kann dann noch genauer aufgedeckt werden, welche Aufrufe im Detail nicht funktionieren.

Der Grundprogrammcode wurde von Herrn Weichand zur Verfügung gestellt. Er wurde von mir so erweitert, dass auch passwortgeschützte WMSe geprüft werden können[85]. Der aktuelle Ist-Stand ist jedoch nicht akzeptabel, da der Test jetzt für alle WMSe 38 Minuten benötigt. Dies ist jedoch die logische Konsequenz des derzeitigen Codes. Wenn für einen WMS zum Beispiel für alle verschiedenen Größen, Formate, Layer, BoundingBoxen und Referenzsysteme eine GetMap-Abfrage getätigt wird, benötigt dies einfach Zeit.

Deswegen muss für die Lösung dieses Problems ein separates Testkonzept entwickelt werden. Es muss hier geprüft werden, ob es wirtschaftlicher ist, den bestehenden Programmcode umzuschreiben oder es besser ist zusätzliche Codes zu entwickeln. Diese Codes könnten als Schnelltests programmiert werden. Das bedeutet, dass zum Beispiel das Referenzsystem vorgegeben wird und nur die anderen Variablen sich ändern.

Ziel sollte es auf jeden Fall sein, die Zeitdauer der automatisierten Tests für GetMap- und GetFeatureInfo-Abfragen zu reduzieren.

[85] Siehe Anhang J

7.2 Test über die GDI-DE Testsuite

Die Testsuite der GDI Deutschland dient dazu, Datensätze und Dienste auf die nationalen und internationalen Standards zu testen[86].

Zukünftig sollen die Dienste des LDBV auch auf die Konformität zu den INSPIRE und OGC-Standards geprüft werden. Das hat den Vorteil, dass nicht die GDI-BY Geschäftsstelle dies tun muss. Außerdem kann so schon während der Realisierung des Dienstes kontrolliert werden, ob er INSPIRE-Konform ist.

Für solch einen Test gibt es schon einen Basiscode[87] von Herrn Sven Böhme, einem Mitarbeiter des Bundesamt für Kartographie und Geodäsie.

7.3 Test von WFS-Diensten

Die WFSe des LDBV und der GDI werden momentan noch nicht von den automatisierten Tests abgedeckt. Dies liegt daran, dass der Programmcode von der OWSLib nicht für passwortgeschützte Dienste geschrieben ist. Deshalb muss er an manchen Stellen umgeschrieben werden.

7.4 Aufbau eines Monitoringsystems

Ein wichtiger Punkt für die Tests ist es ein Monitoring aufzubauen. Durch Jenkins muss die Auswertung noch manuell geschehen bzw. dient das momentane Konzept eher der Erkennung der Fehler und nicht als Frühwarnsystem. Dies soll geändert werden. Dadurch kann das Sachgebiet QM-IuK noch schneller reagieren und muss nicht erst darauf warten, dass beispielsweise ein Kunde auf einen Ausfall eines Dienstes hinweist.

[86] https://testsuite.gdi-de.org/gdi/
[87] https://wiki.gdi-de.org/display/test/Testsuite-API+nutzen, Punkt 7 (Stand 20.05.2016)

8 Fazit

Wenn man sich die nötige Zeit für die Entwicklung eines Konzeptes und die Testdokumente nimmt, kann die Automatisierung von Testverfahren eine große Erleichterung sein. Wobei hier zu beachten ist, dass die Entwicklung nur einen Teil der Arbeit ausmacht. Ständige Pflege und eventuelle Erweiterung und Änderungen der Tests müssen auch getätigt werden.

Durch die Automatisierung dauern die einzelnen Testverfahren der Dienste nicht mehr so lange und können auch schon während der Entwicklung eingesetzt werden. Außerdem sind, im Vergleich zu manuellen Tests, die Ergebnisse des automatisierten Verfahrens wesentlich genauer.

Auch können Änderungen im Back-End[88] ohne ein großes Testverfahren von statten gehen. Durch die Tests kann überprüft werden, ob die Daten geladen werden, oder ob es Probleme an den Schnittstellen gibt.

Alles in allem hat sich die Einführung der automatisierten Tests im Sachgebiet QM-IuK rentiert. Die eingesparte Zeit kann jetzt für andere Dinge genutzt werden. Dadurch wurde die Produktivität des Referats deutlich gesteigert.

[88] Siehe Glossar der Fachwörter

I. Glossar der Fachwörter

Back-End	Als Back-End wird der Teil eines IT-Systems bezeichnet, der sich mit der Datenverarbeitung im Hintergrund beschäftigt.
BayernAtlas	Kartenviewer des LDBV http://geoportal.bayern.de/bayernatlas/
Boolsche Variabel	Eine Variable, die nur zwei Werte annehmen kann
Build	Eng.: to build „bauen“, der aktuelle Stand einer implementierten Software
Capabilities-Viewer	Ein Dienst der GDI. Mit Hilfe des Capablities-Viewer können die Capabilites eines Geodienstes angezeigt werden. https://geoportal.bayern.de/getcapabilities/
Default-Wert	Vorgegebener Wert
Deklaration	Mit dieser wird eine Variable benannt und einem Datentypen zugewiesen.
Exceptions	Eng.: Ausnahme oder Ausnahmesituation, Bezeichnet in der Informatik ein Verfahren, Informationen über einen Fehlerzustand weiterzubehandeln.
Georeferenzierung	Die Zuweisung raumbezogener Informationen zu einem Datensatz.
Integrationstest	Testen mit dem Ziel, Fehlerzustände in den Schnittstellen und im Zusammenspiel zwischen integrierten Komponenten aufzudecken.
Integer	Ein Datentyp für ganze Zahlen

Internet Media Type / Content-Type	Klassifizierung von, über das HTTP übertragenen, Dateien.
Jenkins	ehemals Hudson; ist ein webbasiertes Open Source Continuous Integration System
Jpeg / Jpg	Bildformat
Modultest	Test einer einzelnen Softwareeinheit
Monitoring	Dauerbeobachtung eines bestimmten Systems
(Programm)-Bibliothek	Eine Sammlung von Unterprogrammen, die jeder Zeit in das Hauptprogramm importiert werden können.
Python	Eine dynamische Programmiersprache
Rasterdaten	Datenmodell, in dem Bildinhalte oder räumliche Objekte als quadratische Rasterzellen in einer zweidimensionalen Datenmatrix abgebildet werden.
Shell	Eine einfache Schnittstelle um mit dem Betriebssystem zu kommunizieren.
Styled Layer Descriptor	Ein XML Schema, welches von der OGC definiert wurde.
String	Ein Datentyp für Zeichenketten
UML	Modellierungssprache, eine Sprache zur Beschreibung von Software-Systemen

II. Abkürzungsverzeichnis

ÄDBV	Ämter für Digitalisierung, Breitband und Vermessung
AdV	Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland
BVV	Bayerische Vermessungsverwaltung
BKG	Bundesamt für Kartographie und Geodäsie
CRS	Coordinate reference system (dt.: Koordinatenreferenzsystem)
DOP	Digitales Orthophoto
EPSG	European Petroleum Survey Group Geodesy
GDI	Geodateninfrastruktur
GIS	Geoinformationssystem
GK	Gauß-Krüger-System
GML	Geographic Markup Language (OGC)
HTTP	Hyper Text Transfer Protokoll
ISO	International Organization for Standardization
INSPIRE	Infrastructure for Spatial Information in Europe
IuK	Informations- und Kommunikationstechnik
Jpeg	Joint Photographic Experts Group
KVP	Key-Value Pair
LDBV	Landesamt für Digitalisierung, Breitband und Vermessung
OGC	Open Geospatial Consortium
QM-IuK	Qualitätsmanagement - hier auch Referat 46

REST RESTful	Representational State Transfer
SLD	Styled Layer Descriptor
Smerc	WGS 84 / Pseudo-Mercator-Kachelsatz
SOAP	Simple Object Access Protocol
TC	Technical Committee (dt.: Technisches Komitee)
UML	Unified Modeling Language
UTM	Universal Transversal Mercator System
URL	Uniform Resource Locator
WMS	Web Map Service
WMTS	Web Map Tile Service
WWW	World Wide Web
XML	Extensible Markup Language

Referat 42	Entwicklung Webapplikationen und -dienste, GeodatenOnline, eGovernment
Referat 46	Qualitätsmanagement IuK, Release- und Projektmanagement, QS-ALKIS

III. Abbildungsverzeichnis

IV. Tabellenverzeichnis

V. Quellenverzeichnis

Printmedien

Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland (AdV) (2014): AdV-Profil zum Web Map Tile Service (WMTS), Version 1.0.0.

Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland (AdV) (2012): AdV-Festlegungen zu den INSPIRE Technical Guidance View Services, Version 3.1

Bucsics, Thomas / Baumgartner, Manfred / Seidl, Richard / Gwihs, Stefan (2015): Basiswissen Testautomatisierung: Konzepte, Methoden und Techniken, 2. Auflage, Heidelberg

Clerico, Yvonne / Dr. Scheugenpflug, Stefan / Schleder, Daniela (2009): Arbeitshilfe Geodaten in der Praxis, Version 1.1, München

Drafting Team Metadata and European Commission Joint Research Centre (2008): INSPIRE Metadata Implementing Rules: Technical Guidelines based on EN ISO 19115 and EN ISO 19119, Version 1.0

de La Beaujardière, Jeff (2002): Web Map Service Implementation Specification, Version 1.1.1

Dressmann, Michael / Seifer, Markus (2005): Übersicht der ISO Standards zu Geographischen Informationen / Geomatik, Version 1.03, Potsdam

Flückinger, Urs (SOGI Fachgruppe GIS Technologie) (2005): Interoperabilität für die breite Nutzung von Geodaten: Nationale und internationale Standards

Gumm, Heinz Peter / Sommer, Manfred (2011): Einführung in die Informatik, 9. Auflage, München

Klein, Bernd (2014): Einführung in Python 3: Für Ein- und Umsteiger, 2. Auflage, München

Koordinierungsstelle GDI-NI (2014): Hinweise zum Verwenden von OGC-Webservices, Hannover

Lother, Georg (2012): Vorlesungsscript „Geodateninfrastrukturen“, Hochschule für Angewandte Wissenschaften

Lother, Georg (2012): Vorlesungsscript „Geoinformationssysteme“, Hochschule für Angewandte Wissenschaften

Masó, Joan / Pomakis, Keith / Núria, Julià (2010): ®OpenGIS Web Map Tile Service Implementation Standard, Version 1.0.0

Spillner, Andreas / Linz Tilo (2012): Basiswissen Softwaretest: Aus- und Weiterbildung zum Certified Tester - Foundation Level nach ISTQB-Standard (ISQL-Reihe). 5.Auflage, Heidelberg

The Open Geospatial Consortium (2011): OGC Referenc Model, Version 2.1

Weichand, Jürgen (2014): Moderne Geodatenbereitstellung im Kontext der europäischen Geodateninfrastuktur INSPIRE

Zäch, Reinhard / Feichtner, Astrid / Jud, Michaela / Müller, Julia / Rumpfinger, Hans (2015): Nutzung von Geodatendiensten: Leitfaden, Version 1.0

Internetseiten

http://home.edvsz.fh-osnabrueck.de/skleuker/CSI/Werkzeuge/Jenkins/

http://live.osgeo.org/de/standards/standards.html

http://magic.lib.uconn.edu/help/help_OGC.html

http://spatialreference.org/ref/epsg/31468/

http://www.adv-online.de/Startseite/

http://www.duden.de/rechtschreibung/Norm

http://www.geoinformatik.uni-rostock.de/einzel.asp?ID=982

http://www.iso.org/iso/home.htm

http://www.isotc211.org/

http://www.lesscode.de/initiative/dynamische-programmiersprachen/

http://www.ogcnetwork.net/node/1288

http://www.opengeospatial.org/

http://www.supermap.com/EN/online/iServer%20Java%206R/API/WMTS/WMTS100/wmts1.0.0.htm

https://geopython.github.io/OWSLib/

https://py-tutorial-de.readthedocs.io/de/python-3.3/

https://www.lvermgeo.sachsen-anhalt.de/de/geoservice/wms/main.htm

VI. Anhang

Anhang A: ogc_dop80_oa_base.txt

Imports

```
>>> from __future__ import (absolute_import, division, print_function)
>>> from tests.utils import cast_tuple_int_list, cast_tuple_int_list_srs, resource_file
>>> from owslib.wms import WebMapService

>>> wms = WebMapService('http://www.geodaten.bayern.de/ogc/ogc_dop80_oa.cgi',version='1.1.1')

>>> list(sorted(wms.exceptions))
['application/vnd.ogc.se_blank', 'application/vnd.ogc.se_inimage', 'application/vnd.ogc.se_xml']
```

Layer Aufnahmedatum

BoundingBox

```
>>> wms['by_dop80_info'].boundingBox
(4280000.0, 5220000.0, 4640000.0, 5610000.0, 'EPSG:31468')

>>> wms['by_dop80_info'].boundingBoxWGS84
(8.89292, 47.0828, 13.9782, 50.6269)

>>> list(sorted(wms['by_dop80_info'].crsOptions))
['EPSG:25832', 'EPSG:25833', 'EPSG:31467', 'EPSG:31468', 'EPSG:4258', 'EPSG:4326']
```

GetLegendGraphic

```
>>> sorted(list(wms['by_dop80_info'].styles['default'].values()))
['default', 'http://geodaten.bayern.de/ogc/ogc_dop80_oa.cgi?
version=1.1.1&service=WMS&request=GetLegendGraphic&layer=by_dop80_info&format=image/png']
```

Layer Dop80 Farbe

BoundingBox

```
>>> wms['by_dop80c'].boundingBox
(4280000.0, 5220000.0, 4640000.0, 5610000.0, 'EPSG:31468')
```

>>> wms['by_dop80c'].boundingBoxWGS84

(8.89292, 47.0828, 13.9782, 50.6269)

>>> list(sorted(wms['by_dop80c'].crsOptions))

['EPSG:25832', 'EPSG:25833', 'EPSG:31467', 'EPSG:31468', 'EPSG:4258', 'EPSG:4326']

GetLegendGraphic

>>> sorted(list(wms['by_dop80c'].styles['default'].values()))

['default', 'http://geodaten.bayern.de/ogc/ogc_dop80_oa.cgi?
version=1.1.1&service=WMS&request=GetLegendGraphic&layer=by_dop80c&format=image/png']

LayerDOP 80 (Graustufen)

BoundingBox

>>> wms['by_dop80g'].boundingBox

(4280000.0, 5220000.0, 4640000.0, 5610000.0, 'EPSG:31468')

>>> wms['by_dop80g'].boundingBoxWGS84

(8.89292, 47.0828, 13.9782, 50.6269)

>>> list(sorted(wms['by_dop80g'].crsOptions))

['EPSG:25832', 'EPSG:25833', 'EPSG:31467', 'EPSG:31468', 'EPSG:4258', 'EPSG:4326']

GetLegendGraphic

>>> sorted(list(wms['by_dop80g'].styles['default'].values()))

['default', 'http://geodaten.bayern.de/ogc/ogc_dop80_oa.cgi?
version=1.1.1&service=WMS&request=GetLegendGraphic&layer=by_dop80g&format=image/png']

Layer Copyright

BoundingBox

>>> wms['by_copyright'].boundingBox

(4280000.0, 5220000.0, 4640000.0, 5610000.0, 'EPSG:31468')

>>> wms['by_copyright'].boundingBoxWGS84

(8.89292, 47.0828, 13.9782, 50.6269)

>>> list(sorted(wms['by_copyright'].crsOptions))

['EPSG:25832', 'EPSG:25833', 'EPSG:31467', 'EPSG:31468', 'EPSG:4258', 'EPSG:4326']

GetLegendGraphic

```
>>> wms['by_copyright'].styles
{}
```

GetMap

```
>>> list(sorted(wms.getOperationByName('GetMap').formatOptions))
['image/jpeg', 'image/png']

>>> sorted(list(wms.getOperationByName('GetMap').methods[0].values()))
['Get', 'http://www.geodaten.bayern.de/ogc/ogc_dop80_oa.cgi?']

>>> sorted(list(wms.getOperationByName('GetMap').methods[1].values()))
['Post', 'http://www.geodaten.bayern.de/ogc/ogc_dop80_oa.cgi?']
```

GetCapabilities

```
>>> list(wms.getOperationByName('GetCapabilities').formatOptions)
['application/vnd.ogc.wms_xml']

>>> sorted(list(wms.getOperationByName('GetCapabilities').methods[0].values()))
['Get', 'http://www.geodaten.bayern.de/ogc/ogc_dop80_oa.cgi?']

>>> sorted(list(wms.getOperationByName('GetCapabilities').methods[1].values()))
['Post', 'http://www.geodaten.bayern.de/ogc/ogc_dop80_oa.cgi?']
```

GetFeatureInfo

```
>>> list(wms.getOperationByName('GetFeatureInfo').formatOptions)
['text/html', 'application/vnd.ogc.gml']

>>> sorted(list(wms.getOperationByName('GetFeatureInfo').methods[0].values()))
['Get', 'http://www.geodaten.bayern.de/ogc/ogc_dop80_oa.cgi?']

>>> sorted(list(wms.getOperationByName('GetFeatureInfo').methods[1].values()))
['Post', 'http://www.geodaten.bayern.de/ogc/ogc_dop80_oa.cgi?']

>>> sorted([operation.name for operation in wms.operations]) #letzte Zeile
['GetCapabilities', 'GetFeatureInfo', 'GetLegendGraphic', 'GetMap']
```

Anhang B: ogc_dop80_oa_layer.txt

Imports

```
>>> from __future__ import (absolute_import, division, print_function)
>>> from tests.utils import cast_tuple_int_list, cast_tuple_int_list_srs, resource_file
>>> from owslib.wms import WebMapService
>>> wms = WebMapService('http://www.geodaten.bayern.de/ogc/ogc_dop80_oa.cgi',version='1.1.1')
```

Angaben zu den Layern

```
>>> sorted(list(wms.contents))
['by_copyright', 'by_dop80_info', 'by_dop80c', 'by_dop80g']
```

Layer Aufnahmedatum

```
>>> wms['by_dop80_info'].title
'Aufnahmedatum'
>>> wms['by_dop80_info'].id
'by_dop80_info'
>>> wms['by_dop80_info'].abstract
'Angabe des Aufnahmetages des zugrunde liegenden Luftbildes (Befliegungsdatum) und Nummer des Bildfluges.'
>>> wms['by_dop80_info'].keywords
['Aufnahmetag', 'Aufnahmedatum', 'Befliegungsdatum', 'Bildflug']
```

Dop80 Farbe

```
>>> wms['by_dop80c'].title
'DOP 80 (Farbe)'
>>> wms['by_dop80c'].id
'by_dop80c'
>>> wms['by_dop80c'].abstract
'Die Orthophotos liegen flächendeckend in Farbe vor.'
>>> wms['by_dop80c'].keywords
['Farborthophotos']
```

DOP 80 (Graustufen)

```
>>> wms['by_dop80g'].title
'DOP 80 (Graustufen)'
>>> wms['by_dop80g'].id
'by_dop80g'
```

>>> wms['by_dop80g'].abstract

'Die Orthophotos liegen flächendeckend in Graustufen vor.'

>>> wms['by_dop80g'].keywords

['Graustufenorthophotos']

Copyright

>>> wms['by_copyright'].title

'Copyright'

>>> wms['by_copyright'].id

'by_copyright'

>>> wms['by_copyright'].abstract

'Der Layer dient zur Einblendung des Copyright-Vermerks der Bayerischen Vermessungsverwaltung in der linken oberen Bildecke.'

>>> wms['by_copyright'].keywords

['Copyright', 'Copyright-Vermerk', 'Quellenvermerk', 'Quellenhinweis']

Anhang C: ogc_dop80_oa_prosa.txt

Imports

>>> from __future__ import (absolute_import, division, print_function)

>>> from tests.utils import cast_tuple_int_list, cast_tuple_int_list_srs, resource_file

>>> from owslib.wms import WebMapService

>>> wms = WebMapService('http://www.geodaten.bayern.de/ogc/ogc_dop80_oa.cgi',version='1.1.1')

Beschreibung des Dienstes

>>> wms.identification.type

'WMS_BY_DOP80'

>>> wms.identification.version

'1.1.1'

>>> wms.identification.title

'Digitales Orthophoto 80 cm Bodenauflösung (BVV)'

>>> wms.identification.abstract

'Digitale Orthophotos (DOP) sind vollständig entzerrte, maßstabsgetreue Luftbilder auf Grundlage der Bayernbefliegung, wobei jährlich Eindrittel der bayerischen Landesfläche beflogen wird. Die Bodenpixelgröße des DOP80 beträgt 80 cm ist somit für Darstellungsmaßstäbe von ca. 1 : 8.000 und kleiner geeignet. Bei größeren Darstellungsmaßstäben tritt die grobe Pixelstruktur des Bildes in Erscheinung. Das DOP80 steht über den WMS in Echtfarben (RGB) und in Graustufen zur Verfügung. Weitere Informationen unter: http://www.vermessung.bayern.de/luftbild/bayernbefliegung.html'

>>> list(sorted(wms.identification.keywords))

['BVV', 'Bayerische Vermessungsverwaltung', 'DOP80', 'Digitales Orthophoto', 'OpenData', 'infoMapAccessService']

>>> wms.identification.accessconstraints

'CC-BY vgl. http://creativecommons.org/licenses/by/3.0/de/'

>>> wms.identification.fees

'kostenfrei (mit allen Rechten)'

>>> wms.provider.name

'Landesamt für Digitalisierung, Breitband und Vermessung'

>>> wms.provider.url # fehlendes www wird momentan unterstützt

'http://geodaten.bayern.de'

Kontaktinformationen

>>> wms.provider.contact.organization

'Landesamt für Digitalisierung, Breitband und Vermessung'

>>> wms.provider.contact.name

'Kundenservice'

>>> wms.provider.contact.position

>>> wms.provider.contact.address

'Alexandrastraße 4'

>>> wms.provider.contact.region

'DE-BY'

>>> wms.provider.contact.email

'service@geodaten.bayern.de'

>>> wms.provider.contact.city

'München'

>>> wms.provider.contact.postcode

'80538'

>>> wms.provider.contact.country

'DE'

Anhang D: bayernatlas_wmts_base.txt

```
>>> from __future__ import (absolute_import, division, print_function)
>>> from tests.utils import cast_tuple_int_list, cast_tuple_int_list_srs, resource_file
>>> from owslib.wmts import WebMapTileService

>>> wmts = WebMapTileService('http://www.geodaten.bayern.de/wmts/{ServiceKey}/1.0.0/WMTSCapabili-
ties.xml', version='1.0.0')

Service Metadata URL

>>> wmts.serviceMetadataURL
'http://www.geodaten.bayern.de/wmts/{ServiceKey}/1.0.0/WMTSCapabilities.xml'

TileMatrixSet variants


>>> sorted(wmts.tilematrixsets.keys())
['bvv_gk4', 'smerc']

>>> wmts.tilematrixsets['bvv_gk4'].identifier
'bvv_gk4'

>>> wmts.tilematrixsets['bvv_gk4'].crs
'urn:ogc:def:crs:EPSG::31468'

>>> sorted(wmts.tilematrixsets['bvv_gk4'].tilematrix.keys())
['0', '1', '10', '11', '12', '13', '14', '2', '3', '4', '5', '6', '7', '8', '9']

>>> wmts.tilematrixsets['bvv_gk4'].tilematrix['0'].identifier
'0'

>>> int(wmts.tilematrixsets['bvv_gk4'].tilematrix['0'].scaledenominator)
14628571

>>> cast_tuple_int_list(wmts.tilematrixsets['bvv_gk4'].tilematrix['0'].topleftcorner)
[5924288, 3925712]

>>> wmts.tilematrixsets['bvv_gk4'].tilematrix['0'].tilewidth
256

>>> wmts.tilematrixsets['bvv_gk4'].tilematrix['0'].tileheight
256

>>> wmts.tilematrixsets['bvv_gk4'].tilematrix['0'].matrixwidth
1

>>> wmts.tilematrixsets['bvv_gk4'].tilematrix['0'].matrixheight
1

>>> wmts.tilematrixsets['bvv_gk4'].tilematrix['14'].identifier
'14'

>>> int(wmts.tilematrixsets['bvv_gk4'].tilematrix['14'].scaledenominator)
892

>>> cast_tuple_int_list(wmts.tilematrixsets['bvv_gk4'].tilematrix['14'].topleftcorner)
[5924288, 3925712]
```

```
>>> wmts.tilematrixsets['bvv_gk4'].tilematrix['14'].tilewidth
256

>>> wmts.tilematrixsets['bvv_gk4'].tilematrix['14'].tileheight
256

>>> wmts.tilematrixsets['bvv_gk4'].tilematrix['14'].matrixwidth
16384

>>> wmts.tilematrixsets['bvv_gk4'].tilematrix['14'].matrixheight
16384


>>> wmts.tilematrixsets['smerc'].identifier
'smerc'

>>> wmts.tilematrixsets['smerc'].crs
'urn:ogc:def:crs:EPSG::3857'

>>> sorted(wmts.tilematrixsets['smerc'].tilematrix.keys())
['0', '1', '10', '11', '12', '13', '14', '15', '16', '17', '18', '2', '3', '4', '5', '6', '7', '8', '9']

>>> wmts.tilematrixsets['smerc'].tilematrix['0'].identifier
'0'

>>> int(wmts.tilematrixsets['smerc'].tilematrix['0'].scaledenominator)
559082264

>>> wmts.tilematrixsets['smerc'].tilematrix['0'].tilewidth
256

>>> wmts.tilematrixsets['smerc'].tilematrix['0'].tileheight
256

>>> wmts.tilematrixsets['smerc'].tilematrix['0'].matrixwidth
1

>>> wmts.tilematrixsets['smerc'].tilematrix['0'].matrixheight
1


>>> wmts.tilematrixsets['smerc'].tilematrix['18'].identifier
'18'

>>> wmts.tilematrixsets['smerc'].tilematrix['18'].scaledenominator
2132.729583849784

>>> int(wmts.tilematrixsets['smerc'].tilematrix['18'].scaledenominator)
2132

>>> wmts.tilematrixsets['smerc'].tilematrix['18'].topleftcorner
(-20037508.34278924, 20037508.34278924)

>>> wmts.tilematrixsets['smerc'].tilematrix['18'].tilewidth
256

>>> wmts.tilematrixsets['smerc'].tilematrix['18'].tileheight
256

>>> wmts.tilematrixsets['smerc'].tilematrix['18'].matrixwidth
```

262144

```
>>> wmts.tilematrixsets['smerc'].tilematrix['18'].matrixheight
262144
```

DOP

```
>>> cast_tuple_int_list(wmts['by_dop'].boundingBoxWGS84)
[8, 47, 13, 50]

>>> wmts['by_dop'].styles
{'default': {'isDefault': True}}

>>> wmts['by_dop'].formats
['image/jpeg']

>>> sorted(wmts['by_dop'].tilematrixsetlinks.keys())
['bvv_gk4', 'smerc']
```

Amtliche Karte

```
>>> cast_tuple_int_list(wmts['by_amtl_karte'].boundingBoxWGS84)
[8, 47, 13, 50]

>>> wmts['by_amtl_karte'].styles
{'default': {'isDefault': True}}

>>> wmts['by_amtl_karte'].formats
['image/png']

>>> sorted(wmts['by_amtl_karte'].tilematrixsetlinks.keys())
['bvv_gk4']
```

Beschriftung

```
>>> cast_tuple_int_list(wmts['by_label'].boundingBoxWGS84)
[8, 47, 13, 50]

>>> wmts['by_label'].styles
{'default': {'isDefault': True}}

>>> wmts['by_label'].formats
['image/png']

>>> sorted(wmts['by_label'].tilematrixsetlinks.keys())
['bvv_gk4', 'smerc']
```

Webkarte

```
>>> cast_tuple_int_list(wmts['by_webkarte'].boundingBoxWGS84)
[8, 47, 13, 50]

>>> wmts['by_webkarte'].styles
{'default': {'isDefault': True}}

>>> wmts['by_webkarte'].formats
['image/png']

>>> sorted(wmts['by_webkarte'].tilematrixsetlinks.keys())
['bvv_gk4', 'smerc']
```

Webkarte

```
>>> cast_tuple_int_list(wmts['by_webkarte_grau'].boundingBoxWGS84)
[8, 47, 13, 50]

>>> wmts['by_webkarte_grau'].styles
{'default': {'isDefault': True}}

>>> wmts['by_webkarte_grau'].formats
['image/png']

>>> sorted(wmts['by_webkarte_grau'].tilematrixsetlinks.keys())
['bvv_gk4', 'smerc']
```

Operations

```
>>> list(sorted([op.name for op in wmts.operations]))
[]

>>> tile = wmts.gettile(layer='by_amtl_karte', tilematrixset='bvv_gk4', tilematrix='4', row='8', column='8',
format="image/png")
>>> out = open(scratch_file('wmts_amtl_karte.png'), 'wb')
>>> bytes_written = out.write(tile.read())
>>> out.close()



>>> tile = wmts.gettile(layer='by_webkarte', tilematrixset='bvv_gk4', tilematrix='4', row='8', column='8',
format="image/png")
>>> out = open(scratch_file('wmts_webkarte.png'), 'wb')
>>> bytes_written = out.write(tile.read())
>>> out.close()



>>> tile = wmts.gettile(layer='by_webkarte_grau', tilematrixset='bvv_gk4', tilematrix='4', row='8', column='8',
format="image/png")
>>> out = open(scratch_file('wmts_webkarte_g.png'), 'wb')
>>> bytes_written = out.write(tile.read())
>>> out.close()



>>> tile = wmts.gettile(layer='by_dop', tilematrixset='bvv_gk4', tilematrix='4', row='8', column='8',
format="image/png")
>>> out = open(scratch_file('wmts_dop.png'), 'wb')
>>> bytes_written = out.write(tile.read())
>>> out.close()
```

Anhang E: bayernatlas_wmts_layer.txt

```
>>> from __future__ import (absolute_import, division, print_function)
>>> from tests.utils import cast_tuple_int_list, cast_tuple_int_list_srs, resource_file
>>> from owslib.wmts import WebMapTileService

>>> wmts = WebMapTileService('http://www.geodaten.bayern.de/wmts/{ServiceKey}/1.0.0/WMTSCapabili-
ties.xml', version='1.0.0')


Content Layers

>>> sorted(list(wmts.contents))
['by_amtl_karte', 'by_dop', 'by_label', 'by_webkarte', 'by_webkarte_grau']


DOP

>>> wmts['by_dop'].title
'Luftbild Bayern'

>>> wmts['by_dop'].id
'by_dop'

>>> wmts['by_dop'].infoformats
[]


Amtliche Karte


>>> wmts['by_amtl_karte'].title
'Amtliche Karte Bayern'

>>> wmts['by_amtl_karte'].id
'by_amtl_karte'

>>> wmts['by_amtl_karte'].infoformats
[]


Beschriftung

>>> wmts['by_label'].title
'Beschriftungen Bayern'

>>> wmts['by_label'].id
'by_label'

>>> wmts['by_label'].infoformats
[]


Webkarte

>>> wmts['by_webkarte'].title
'Webkarte Bayern'

>>> wmts['by_webkarte'].id
```

'by_webkarte'

```
>>> wmts['by_webkarte'].infoformats
[]
```

Webkarte Grau

```
>>> wmts['by_webkarte_grau'].title
'Webkarte S/W Bayern'

>>> wmts['by_webkarte_grau'].id
'by_webkarte_grau'

>>> wmts['by_webkarte_grau'].infoformats
[]
```

Anhang F: bayernatlas_wmts_prosa.txt

```
>>> from __future__ import (absolute_import, division, print_function)
>>> from tests.utils import cast_tuple_int_list, cast_tuple_int_list_srs, resource_file
>>> from owslib.wmts import WebMapTileService

>>> wmts = WebMapTileService('http://www.geodaten.bayern.de/wmts/{ServiceKey}/1.0.0/WMTSCapabili-
ties.xml', version='1.0.0')
```

Capabilities

```
>>> wmts.identification.type
'OGC:WMTS'

>>> wmts.identification.version
'1.0.0'

>>> wmts.identification.title
'BayernAtlas-WMTS (beta-Version)'

>>> wmts.identification.abstract
'Der BayernAtlas-WMTS in der beta-Version der Bayerischen Vermessungverwaltung übermittelt die Hinter-
grunddarstellungen des BayernAtlas in vordefinierten Rasterkacheln. Der Dienst enthält die Layer Webkarte Bay-
ern, Webkarte S/W Bayern, Luftbild Bayern, Amtliche Karte Bayern sowie einen Beschriftungslayer. Die Raster-
kacheln sind in den Koordinatensystemen WGS84 Pseudo-Merkator (EPSG:3857) und Gauß-Krüger 12°-Streifen
(EPSG:31468) abrufbar. Bitte beachten Sie, dass die Karteninhalte in den Layern noch geändert werden können.'

>>> wmts.identification.keywords
['infoMapAccessService', 'Geobasisdaten', 'WebAtlasDE', 'Webkarte', 'Topographische Karten', 'Parzellarkarte',
'Orthophotos', 'Luftbilder', 'WMTS', 'TMS', 'Bayern', 'Bayerische Vermessungsverwaltung', 'BVV']
>>> wmts.identification.accessconstraints
'Für die Nutzung des Dienstes wird ein Service-Key benötigt. Es gelten die Nutzungsbedingungen (http://vermes-
sung.bayern.de/service/Nutzungshinweise/nutzungsbedingungen.html) der Bayerischen Vermessungsverwaltung.
Des Weiteren gelten für Fremddaten, die in den Kachelarchiven verwendet werden, die Attributionen des jeweili-
gen Datenlieferanten. Eine Liste der geforderten Attributionen finden Sie unter http://www.geodaten.bayern.de/ba-
data/WMTS_Layer_Attributionen.pdf.'
>>> wmts.identification.fees
'Es gilt die Gebühren- und Preisliste (GebPL) (siehe http://vermessung.bayern.de/file/pdf/1269/Geb%C3%BCh-
ren_und_Preisliste.pdf) der Bayerischen Vermessungsverwaltung.'
```

Service Provider

```
>>> wmts.provider.name
'Landesamt für Digitalisierung, Breitband und Vermessung'

>>> wmts.provider.url
'http://www.geodaten.bayern.de'
```

Kontaktdaten

```
>>> wmts.provider.contact.name
'Kundenservice'

>>> wmts.provider.contact.position
'Ansprechpartner'

>>> wmts.provider.contact.phone
```

'+49-89-2129-1111'

>>> wmts.provider.contact.fax
'+49-89-2129-1113'

>>> wmts.provider.contact.address
'Alexandrastraße 4'

>>> wmts.provider.contact.region
'DE-BY'

>>> wmts.provider.contact.email
'service@geodaten.bayern.de'

>>> wmts.provider.contact.city
'München'

>>> wmts.provider.contact.postcode
'80538'

>>> wmts.provider.contact.country
'DE'

>>> wmts.provider.contact.hours
'Mo.-Do. 8-16 Uhr und Fr. 8-14 Uhr'

>>> wmts.provider.contact.instructions
'Bei Fragen zu den Geodaten und -diensten der Bayerischen Vermessungsverwaltung können Sie uns während der Sprechzeiten anrufen. Unsere Mitarbeiterinnen und Mitarbeiter beraten Sie gern.'

Anhang G: Jenkins Report vom 12.04.2016

Report generated on 12-Apr-2016 at 14:02:14

Environment

Platform	Linux-3.13.0-79-generic-x86_64-with-debian-7.9
Python	3.3.6

Summary

119 tests ran in 31.91 seconds.
111 passed, 1 skipped, 8 failed, 0 errors.
0 expected failures, 0 unexpected passes.

Results

Result	*Test*	*Duration*	
Passed	csw/gdi/external/csw_gdi-by_gdi-de.txt	0.91	No log output captured.
Passed	wms/external/ALE_DFK/ogc_ale_dfk_base.txt	0.21	No log output captured.
Passed	wms/external/ALE_DFK/ogc_ale_dfk_layer.txt	0.18	No log output captured.
Failed	wms/external/ALE_DFK/ogc_ale_dfk_prosa.txt	0.21	028 029 030 >>> wms.provider.name Expected: 'Landesamt für Digitalisierung, Breitband und Vermessung' Got: 'Landesamt fuer Vermessung und Geoinformation' /mount/tests/wms/external/ALE_DFK/ogc_ale_dfk_prosa.txt:30: DocTestFailure
Passed	wms/external/ATKIS/ogc_atkis_dlm25_base.txt	0.20	No log output captured.
Failed	wms/external/ATKIS/ogc_atkis_dlm25_layer.txt	0.20	012 013 >>> sorted(list(wms.contents)) Expected: ['by_basisdlm_gebiete', 'by_basisdlm_gewaesser_relief', 'by_basisdlm_komplett', 'by_basisdlm_ohne_texte', 'by_basisdlm_siedlung', 'by_basisdlm_texte', 'by_basisdlm_vegetation', 'by_basisdlm_verkehr'] Got: ['WMS_BY_ATKIS_DLM', 'by_basisdlm_gebiete', 'by_basisdlm_gewaesser_relief', 'by_basisdlm_komplett', 'by_basisdlm_ohne_texte', 'by_basisdlm_siedlung', 'by_basisdlm_texte', 'by_basisdlm_vegetation', 'by_basisdlm_verkehr'] /mount/tests/wms/external/ATKIS/ogc_atkis_dlm25_layer.txt:13: DocTestFailure
Passed	wms/external/ATKIS/ogc_atkis_dlm25_prosa.txt	0.18	No log output captured.
Passed	wms/external/BOSCH/ogc_alkis_bosch_base.txt	0.16	No log output captured.
Passed	wms/external/BOSCH/ogc_alkis_bosch_layer.txt	0.19	No log output captured.

Result	*Test*	*Duration*	
Passed	wms/external/BOSCH/ogc_alkis_bosch_prosa.txt	0.16	No log output captured.
Passed	wms/external/DFK/ogc_dfk_base.txt	0.22	No log output captured.
Passed	wms/external/DFK/ogc_dfk_layer.txt	0.20	No log output captured.
Passed	wms/external/DFK/ogc_dfk_prosa.txt	0.21	No log output captured.
Passed	wms/external/DOK/ogc_dok_base.txt	0.24	No log output captured.
Passed	wms/external/DOK/ogc_dok_layer.txt	0.17	No log output captured.
Passed	wms/external/DOK/ogc_dok_prosa.txt	0.20	No log output captured.
Passed	wms/external/DOP20/ogc_dop20_base.txt	0.25	No log output captured.
Passed	wms/external/DOP20/ogc_dop20_layer.txt	0.15	No log output captured.
Passed	wms/external/DOP20/ogc_dop20_prosa.txt	0.16	No log output captured.
Passed	wms/external/DOP200/ogc_dop200_oa_base.txt	0.05	No log output captured.
Passed	wms/external/DOP200/ogc_dop200_oa_layer.txt	0.04	No log output captured.
Passed	wms/external/DOP200/ogc_dop200_oa_prosa.txt	0.04	No log output captured.
Passed	wms/external/DOP20_CIR/ogc_dop20_cir_base.txt	0.17	No log output captured.
Passed	wms/external/DOP20_CIR/ogc_dop20_cir_layer.txt	0.21	No log output captured.
Passed	wms/external/DOP20_CIR/ogc_dop20_cir_prosa.txt	0.39	No log output captured.
Passed	wms/external/DOP40/ogc_dop40_base.txt	0.16	No log output captured.
Passed	wms/external/DOP40/ogc_dop40_layer.txt	0.18	No log output captured.
Passed	wms/external/DOP40/ogc_dop40_prosa.txt	0.15	No log output captured.
Passed	wms/external/DOP40_CIR/ogc_dop40_cir_base.txt	0.16	No log output captured.
Passed	wms/external/DOP40_CIR/ogc_dop40_cir_layer.txt	0.14	No log output captured.
Passed	wms/external/DOP40_CIR/ogc_dop40_cir_prosa.txt	0.14	No log output captured.
Passed	wms/external/DOP80/ogc_dop80_oa-funktion.txt	19.67	No log output captured.
Skipped	wms/external/DOP80/ogc_dop80_oa-test-suite.txt	0.00	('/usr/local/lib/python3.3/site-packages/_pytest/doctest.py', 165, 'Skipped: all tests skipped by +SKIP option')
Passed	wms/external/DOP80/ogc_dop80_oa_base.txt	0.04	No log output captured.
Passed	wms/external/DOP80/ogc_dop80_oa_layer.txt	0.03	No log output captured.
Passed	wms/external/DOP80/ogc_dop80_oa_prosa.txt	0.04	No log output captured.
Passed	wms/external/DTK100/ogc_dtk100_base.txt	0.15	No log output captured.
Passed	wms/external/DTK100/ogc_dtk100_layer.txt	0.16	No log output captured.
Passed	wms/external/DTK100/ogc_dtk100_prosa.txt	0.14	No log output captured.
Passed	wms/external/DTK25/ogc_dtk25_base.txt	0.14	No log output captured.
Passed	wms/external/DTK25/ogc_dtk25_layer.txt	0.14	No log output captured.
Passed	wms/external/DTK25/ogc_dtk25_prosa.txt	0.14	No log output captured.
Passed	wms/external/DTK50/ogc_dtk50_base.txt	0.14	No log output captured.
Passed	wms/external/DTK50/ogc_dtk50_layer.txt	0.13	No log output captured.

Result	*Test*	*Duration*	
Passed	wms/external/DTK50/ogc_dtk50_prosa.txt	0.14	No log output captured.
Passed	wms/external/DTK500/ogc_dtk500_oa_base.txt	0.24	No log output captured.
Passed	wms/external/DTK500/ogc_dtk500_oa_layer.txt	0.03	No log output captured.
Passed	wms/external/DTK500/ogc_dtk500_oa_prosa.txt	0.03	No log output captured.
Passed	wms/external/Freizeitwege/ogc_fzw_oa_base.txt	0.06	No log output captured.
Passed	wms/external/Freizeitwege/ogc_fzw_oa_layer.txt	0.04	No log output captured.
Passed	wms/external/Freizeitwege/ogc_fzw_oa_prosa.txt	0.04	No log output captured.
Passed	wms/external/Hoehenlinien/ogc_dhk_base.txt	0.17	No log output captured.
Passed	wms/external/Hoehenlinien/ogc_dhk_layer.txt	0.13	No log output captured.
Passed	wms/external/Hoehenlinien/ogc_dhk_prosa.txt	0.15	No log output captured.
Passed	wms/external/Luftbilder_Alliierte/ogc_luftbilder_1941-1945_base.txt	0.04	No log output captured.
Passed	wms/external/Luftbilder_Alliierte/ogc_luftbilder_1941-1945_layer.txt	0.03	No log output captured.
Passed	wms/external/Luftbilder_Alliierte/ogc_luftbilder_1941-1945_prosa.txt	0.03	No log output captured.
Passed	wms/external/Schummerung/ogc_schummerung_base.txt	0.15	No log output captured.
Passed	wms/external/Schummerung/ogc_schummerung_layer.txt	0.14	No log output captured.
Passed	wms/external/Schummerung/ogc_schummerung_prosa.txt	0.15	No log output captured.
Failed	wms/external/Verwaltungsgrenzen_DFK_Basis/ogc_verwgrenzen_base.txt	0.16	No log output captured.
Passed	wms/external/Verwaltungsgrenzen_DFK_Basis/ogc_verwgrenzen_layer.txt	0.16	No log output captured.
Passed	wms/external/Verwaltungsgrenzen_DFK_Basis/ogc_verwgrenzen_prosa.txt	0.14	No log output captured.
Passed	wms/internal/ALE_DFK/ogc_ale_dfk_base.txt	0.10	No log output captured.
Passed	wms/internal/ALE_DFK/ogc_ale_dfk_layer.txt	0.10	No log output captured.
Passed	wms/internal/ALE_DFK/ogc_ale_dfk_prosa.txt	0.09	No log output captured.
Passed	wms/internal/ALKIS-TN/ogc_alkis_tn_base.txt	0.03	No log output captured.
Passed	wms/internal/ALKIS-TN/ogc_alkis_tn_layer.txt	0.02	No log output captured.
Passed	wms/internal/ALKIS-TN/ogc_alkis_tn_prosa.txt	0.02	No log output captured.
Passed	wms/internal/ATKIS/ogc_atkis_dlm25_base.txt	0.03	No log output captured.

Result	*Test*	*Duration*	
Failed	wms/internal/ATKIS/ogc_atkis_dlm25_layer.txt	0.02	012 013 >>> sorted(list(wms.contents)) Expected: ['by_basisdlm_gebiete', 'by_basisdlm_gewaesser_relief', 'by_basisdlm_komplett', 'by_basisdlm_ohne_texte', 'by_basisdlm_siedlung', 'by_basisdlm_texte', 'by_basisdlm_vegetation', 'by_basisdlm_verkehr'] Got: ['WMS_BY_ATKIS_DLM', 'by_basisdlm_gebiete', 'by_basisdlm_gewaesser_relief', 'by_basisdlm_komplett', 'by_basisdlm_ohne_texte', 'by_basisdlm_siedlung', 'by_basisdlm_texte', 'by_basisdlm_vegetation', 'by_basisdlm_verkehr'] /mount/tests/wms/internal/ATKIS/ogc_atkis_dlm25_layer.txt:13: DocTestFailure
Passed	wms/internal/ATKIS/ogc_atkis_dlm25_prosa.txt	0.02	No log output captured.
Passed	wms/internal/BOSCH/ogc_alkis_bosch_base.txt	0.02	No log output captured.
Passed	wms/internal/BOSCH/ogc_alkis_bosch_layer.txt	0.02	No log output captured.
Passed	wms/internal/BOSCH/ogc_alkis_bosch_prosa.txt	0.02	No log output captured.
Passed	wms/internal/DFK/ogc_dfk_base.txt	0.12	No log output captured.
Passed	wms/internal/DFK/ogc_dfk_layer.txt	0.10	No log output captured.
Passed	wms/internal/DFK/ogc_dfk_prosa.txt	0.09	No log output captured.
Passed	wms/internal/DOK/ogc_dok_base.txt	0.02	No log output captured.
Passed	wms/internal/DOK/ogc_dok_layer.txt	0.02	No log output captured.
Passed	wms/internal/DOK/ogc_dok_prosa.txt	0.02	No log output captured.
Passed	wms/internal/DOP20/ogc_dop20_base.txt	0.02	No log output captured.
Failed	wms/internal/DOP20/ogc_dop20_layer.txt	0.02	011 012 >>> sorted(list(wms.contents)) Expected: ['adv_copyright', 'adv_dop20c', 'adv_dop20g', 'adv_md_dop20'] Got: ['OGC:WMS', 'adv_copyright', 'adv_dop20c', 'adv_dop20g', 'adv_md_dop20'] /mount/tests/wms/internal/DOP20/ogc_dop20_layer.txt:12: DocTestFailure
Passed	wms/internal/DOP20/ogc_dop20_prosa.txt	0.02	No log output captured.
Passed	wms/internal/DOP20_CIR/ogc_dop20_cir_base.txt	0.02	No log output captured.
Failed	wms/internal/DOP20_CIR/ogc_dop20_cir_layer.txt	0.02	009 Angaben zu den Layern 010 011 >>> sorted(list(wms.contents)) Expected: ['adv_copyright', 'adv_dop20_cir', 'adv_md_dop20_cir'] Got: ['OGC:WMS','adv_copyright', 'adv_dop20_cir', 'adv_md_dop20_cir'] /mount/tests/wms/internal/DOP20_CIR/ogc_dop20_cir_layer.txt:11: DocTestFailure
Passed	wms/internal/DOP20_CIR/ogc_dop20_cir_prosa.txt	0.02	No log output captured.
Passed	wms/internal/DOP40/ogc_dop40_base.txt	0.02	No log output captured.

Result	*Test*	*Duration*	
Failed	wms/internal/DOP40/ogc_dop40_layer.txt	0.02	011 012 >>> sorted(list(wms.contents)) Expected: ['adv_copyright', 'adv_dop40c', 'adv_dop40g', 'adv_md_dop40'] Got: ['OGC:WMS', 'adv_copyright', 'adv_dop40c', 'adv_dop40g', 'adv_md_dop40'] /mount/tests/wms/internal/DOP40/ogc_dop40_layer.txt:12: DocTestFailure
Passed	wms/internal/DOP40/ogc_dop40_prosa.txt	0.02	No log output captured.
Passed	wms/internal/DOP40_CIR/ogc_dop40_cir_base.txt	0.02	No log output captured.
Failed	wms/internal/DOP40_CIR/ogc_dop40_cir_layer.txt	0.02	012 013 >>> sorted(list(wms.contents)) Expected: ['adv_copyright', 'adv_dop40_cir', 'adv_md_dop40_cir'] Got: ['OGC:WMS', 'adv_copyright', 'adv_dop40_cir', 'adv_md_dop40_cir'] /mount/tests/wms/internal/DOP40_CIR/ogc_dop40_cir_layer.txt:13: DocTestFailure
Passed	wms/internal/DOP40_CIR/ogc_dop40_cir_prosa.txt	0.02	No log output captured.
Passed	wms/internal/DTK100/ogc_dtk100_base.txt	0.02	No log output captured.
Passed	wms/internal/DTK100/ogc_dtk100_layer.txt	0.02	No log output captured.
Passed	wms/internal/DTK100/ogc_dtk100_prosa.txt	0.02	No log output captured.
Passed	wms/internal/DTK25/ogc_dtk25_base.txt	0.02	No log output captured.
Passed	wms/internal/DTK25/ogc_dtk25_layer.txt	0.02	No log output captured.
Passed	wms/internal/DTK25/ogc_dtk25_prosa.txt	0.02	No log output captured.
Passed	wms/internal/DTK50/ogc_dtk50_base.txt	0.02	No log output captured.
Passed	wms/internal/DTK50/ogc_dtk50_layer.txt	0.02	No log output captured.
Passed	wms/internal/DTK50/ogc_dtk50_prosa.txt	0.02	No log output captured.
Passed	wms/internal/Hoehenlinien/ogc_dhk_base.txt	0.02	No log output captured.
Passed	wms/internal/Hoehenlinien/ogc_dhk_layer.txt	0.14	No log output captured.
Passed	wms/internal/Hoehenlinien/ogc_dhk_prosa.txt	0.14	No log output captured.
Passed	wms/internal/Schummerung_(mit_GetFeatureInfo)/ogc_schummerung_plus_base.txt	0.02	No log output captured.
Passed	wms/internal/Schummerung_(mit_GetFeatureInfo)/ogc_schummerung_plus_layer.txt	0.02	No log output captured.
Passed	wms/internal/Schummerung_(mit_GetFeatureInfo)/ogc_schummerung_plus_prosa.txt	0.02	No log output captured.
Passed	wms/internal/Schummerung_(ohne_GetFeatureInfo)/ogc_schummerung_base.txt	0.02	No log output captured.
Passed	wms/internal/Schummerung_(ohne_GetFeatureInfo)/ogc_schummerung_layer.txt	0.02	No log output captured.
Passed	wms/internal/Schummerung_(ohne_GetFeatureInfo)/ogc_schummerung_prosa.txt	0.02	No log output captured.

Result	*Test*	*Duration*	
Failed	wms/internal/Verwaltungsgrenzen_DFK_Basis/ogc_verwgrenzen_base.txt	0.03	No log output captured.
Failed	wms/internal/Verwaltungsgrenzen_DFK_Basis/ogc_verwgrenzen_layer.txt	0.03	012 013 >>> sorted(list(wms.contents)) Expected: ['adv_copyright', 'adv_vwg_by_bezeichnung', 'adv_vwg_by_grenze', 'adv_vwg_by_komplett', 'adv_vwg_gde_bezeichnung', 'adv_vwg_gde_grenze', 'adv_vwg_gde_komplett', 'adv_vwg_gmkg_bezeichnung', 'adv_vwg_gmkg_grenze', 'adv_vwg_gmkg_komplett', 'adv_vwg_lkr_bezeichnung', 'adv_vwg_lkr_grenze', 'adv_vwg_lkr_komplett', 'adv_vwg_rbz_bezeichnung', 'adv_vwg_rbz_grenze', 'adv_vwg_rbz_komplett'] Got: ['adv_copyright', 'adv_vwg_by_bezeichnung', 'adv_vwg_by_grenze', 'adv_vwg_by_komplett', 'adv_vwg_gde_bezeichnung', 'adv_vwg_gde_grenze', 'adv_vwg_gde_komplett', 'adv_vwg_gmkg_bezeichnung', 'adv_vwg_gmkg_grenze', 'adv_vwg_gmkg_komplett', 'adv_vwg_lkr_bezeichnung', 'adv_vwg_lkr_grenze', 'adv_vwg_lkr_komplett', 'adv_vwg_rbz_bezeichnung', 'adv_vwg_rbz_grenze', 'adv_vwg_rbz_komplett', 'verwaltungsgrenzen'] /mount/tests/wms/internal/Verwaltungsgrenzen_DFK_Basis/ogc_verwgrenzen_layer.txt:13: DocTestFailure
Passed	wms/internal/Verwaltungsgrenzen_DFK_Basis/ogc_verwgrenzen_prosa.txt	0.02	No log output captured.
Passed	wmts/internal/Bayernatlas/bayernatlas_wmts_base.txt	0.08	No log output captured.
Passed	wmts/internal/Bayernatlas/bayernatlas_wmts_get_tile_amtl_karte_gk4.txt	0.21	No log output captured.
Passed	wmts/internal/Bayernatlas/bayernatlas_wmts_get_tile_dop_gk4.txt	0.15	No log output captured.
Passed	wmts/internal/Bayernatlas/bayernatlas_wmts_layer.txt	0.08	No log output captured.
Passed	wmts/internal/Bayernatlas/bayernatlas_wmts_prosa.txt	0.06	No log output captured.
Passed	wmts/internal/Example/wmts_weichand_beispiel.txt	0.19	No log output captured.

Anhang H: TileMatrix 4 für das GK4-Koordinateneferenzsystem des BayernAtlas WMTS

```
−<TileMatrix>
    <ows:Title>Zoom Level 4 (256 m/pixel)</ows:Title>
    <ows:Identifier>4</ows:Identifier>
    <ScaleDenominator>914285.71</ScaleDenominator>
    <TopLeftCorner>5924288 3925712</TopLeftCorner>
    <TileWidth>256</TileWidth>
    <TileHeight>256</TileHeight>
    <MatrixWidth>16</MatrixWidth>
    <MatrixHeight>16</MatrixHeight>
  </TileMatrix>
```

Anhang I: Vetriebskonzept BayerAtlas-WMTS DOP Layer: Zoomstufen

GK4		Web Mercator	
Zoom	BA [m]	Zoom	BA [m]
		0	104755,2
		1	52377,6
		2	26188,8
		3	13094,4
		4	6547,2
0	4096	5	3273,6
1	2048	6	1636,8
2	1024	7	818,4
3	512	8	409,2
4	256	9	204,8
5	128	10	102,4
6	64	11	51,2
7	32	12	25,6
8	16	13	12,8
9	8	14	6,4
10	4	15	3,2
11	2	16	1,6
12	1	17	0,8
13	0,5	18	0,4
14	0,25	19	0,2

Anhang J: Programmcode zum testen von GetFeatureInfo und GetMap

```
__author__ = 'weich_ju'
__author__ = 'muell_ch'

import logging
import sys
owslib_log = logging.getLogger('owslib')
owslib_log.setLevel(logging.DEBUG)
logging.basicConfig(stream=sys.stdout, level=logging.ERROR)
from owslib.wms import WebMapService
from owslib.wms import ServiceException
from pyproj import Proj, transform
from io import BytesIO
from PIL import Image

# Transformation der BoundingBox über pyproj

def transform_bbox(s_srs, t_srs, bbox):
    s_prj = Proj(init=s_srs)
    t_prj = Proj(init=t_srs)
    s_ulx = bbox[0]
    s_uly = bbox[1]
    t_ulx, t_uly = transform(s_prj,t_prj ,s_ulx, s_uly)
    s_orx = bbox[2]
    s_ory = bbox[3]
    t_orx, t_ory = transform(s_prj,t_prj ,s_orx, s_ory)
    return (t_ulx, t_uly, t_orx, t_ory)

# Test in allen Kombinationen (alle Layer, alle Formate, alle
Koordinatenreferenzsysteme - GetMap + GetFeatureInfo)

def run_test(url):
    wms = WebMapService(url,version='1.1.1', username='[redacted]', password='[redacted]')
    for operation in wms.operations:
        default_srs = 'EPSG:31468'
        bbox = (4500000,5500000,4500500,5500500)
        size = (500, 500)
        for format in operation.formatOptions:
                for layer in wms.contents:
                    for srs in wms[layer].crsOptions:
                        t_bbox = None
                        if not srs == default_srs:
                            t_bbox = transform_bbox(default_srs, srs, bbox)
                        else:
                            t_bbox = bbox
                        if operation.name == 'GetMap':
                            try:
                                response = wms.getmap(layers=[layer], srs=srs,
bbox=t_bbox, size=size, format=format)
                                for pil_format in ['jpg', 'jpeg', 'png', 'tif']:
                                    if pil_format in format:
                                        img =
      Image.open(BytesIO(response.read()))
                            except ServiceException as e:
```

```
                                print(e.message)
                        if operation.name == 'GetFeatureInfo':
                            if wms[layer].queryable:
                                try:
                                    img_format=
wms.getOperationByName('GetMap').formatOptions[0]
                                    response =
wms.getfeatureinfo(layers=[layer], srs=srs, bbox=t_bbox, size=size,
format=img_format, info_format=format, xy=(250,250))
                                except ServiceException as e:
                                    print(e.message)


if __name__ == "__main__":
    run_test('http://www.geodaten.bayern.de/ogc/ogc_dop80_oa.cgi')
```

VII. Erklärung

Erklärung gemäß § 15 Abs. 5 APO in Zusammenhang mit § 35 Abs. 7 RaPO

Name:	Müller-Wilke
Vorname:	Christina
Geburtsdatum:	13.03.1991
Studiengang:	Geoinformatik und Satellitenpositionierung
Studiengruppe:	7W
Matrikel- Nr.:	03472712
~~Winter- /~~Sommersemester:	2016
Betreuer~~/in~~:	Prof. Dr. Georg Lother

Hiermit erkläre ich, dass ich die Bachelorarbeit selbständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel benützt, sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Krailling, _____ _____

Ort, den Datum des Abgabetermins Unterschrift