Prototyp des Geobrowsers (M 1.4.3.1)

Version 04/22/2013
Arbeitspaket 1.4
Verantwortlicher Partner Technische Universität Darmstadt

DARIAH-DE
Aufbau von Forschungsinfrastrukturen für die e-Humanities

Dieses Forschungs- und Entwicklungsprojekt wird / wurde mit Mitteln des Bundesministeriums für Bildung und Forschung (BMBF), Förderkennzeichen 01UG1110A bis M, gefördert und vom Projektträger im Deutschen Zentrum für Luft- und Raumfahrt (PT-DLR) betreut.

GEFÖRDERT VOM

Bundesministerium für Bildung und Forschung

Projekt: DARIAH-DE: Aufbau von Forschungsinfrastrukturen für die e-Humanities
BMBF Förderkennzeichen: 01UG1110A bis M
Laufzeit: März 2011 bis Februar 2014

Dokumentstatus: Final
Verfügbarkeit: Öffentlich
Autoren:

Stefan E. Funk (SUB Göttingen)
Ubbo Veentjer (SUB Göttingen)
Richard Eckart de Castilho (Technische Universität Darmstadt)

Revisionsverlauf:

Datum	Autor	Kommentare
31.08.2012	Stefan E. Funk, Ubbo Veentjer	Programmierung, Dokumentation
03.09.2012	Richard Eckart de Castilho	Verfügbare Informationen aus dem Wiki übernommen.
22.04.2013	Stefan E. Funk	GeoBrowser Wiki-Seiten als DOC angehängt, Kommentare von Christiane eingearbeitet, um Datasheet Editor ergänzt

Inhaltsverzeichnis:

1. Zusammenfassung des Meilensteins

Der Prototyp des DARIAH-DE Geobrowsers wurde erfolgreich fertig gestellt. Er ist erreichbar unter der URL

http://dev2.dariah.eu/e4d

Der Prototyp wurde durch Anpassungen an verschiedenen Softwarepaketen realisiert. Diese Anpassungen, sowie eine Installationsanleitung sind im DARIAH-DE-Wiki dokumentiert (siehe Anhang 1):

https://dev2.dariah.eu/wiki/pages/viewpage.action?pageId=8128253

Die Installation des Prototyps für DARIAH-DE als Dienst ist ebenfalls entsprechend der Vorlage zur Dokumentation von Diensten im Wiki dokumentiert (siehe Anhang 2):

https://dev2.dariah.eu/wiki/display/DARIAHDE/GeoBrowser

Der Quellcode des DARIAH Geobrowsers kann aus dem Subversion Repository abgerufen werden:

http://dev.dariah.eu/svn/repos/eu.dariah.de/ap1/sti-gwt-dariah-geobrowser

Während der Entwicklung wurde das DARIAH Jira verwendet um Aufgaben und Benutzerfeedback zu verwalten. Das System kann auch weiterhin dazu verwendet werden um Rückmeldungen und Fehlerbeschreibungen zu liefern. Werden weitere Entwicklungen am Prototypen vorgenommen, so werden diese ebenfalls dort als Aufgaben geführt.

https://dev.dariah.eu/jira/browse/GEOB

Außerdem wurde die Funktionalität des GeoBrowsers um einen Datasheet Editor erweitert, den man unter folgender URL erreichen kann. Er ermöglicht eine Bearbeitung der Daten per Tabelle in einem Browser-Fenster, die Festlegung von Geo-Koordinaten mit dort eingebundenen Karten und automatischer Ermittlung von Ortsnamen.

http://ref.dariah.eu/workflow/

Der Prototyp wurde umgesetzt von Stefan E. Funk (SUB Göttingen), Ubbo Veentjer (SUB Göttingen)

Tests wurden durchgeführt von: Beata Mache (STI Essen) und Thomas Kollatz (STI Essen)

Die Produktivversion des GeoBrowsers wird in Monat 36 als Meilenstein M1.4.3.2 der Öffentlichkeit zur Verfügung gestellt werden.

2. Anhang 1 – GeoBrowser Wiki-Seite (Stand 22.04.2013)

2.1. GeoBrowser (e4D)

2.2. Technische Vorraussetzungen

Die technischen Vorraussetzungen werden im Rahmen der Paketverwaltung für SLES 11 (oder der von anderen Linuxdistributionen) bereitgestellt. Für SLES kann es notwendig sein das SDK als Paketquelle hinzuzufügen.

2.2.1. Betrieb

- Oracle Java [1]
- Apache Tomcat [2]

2.2.2. Kompilierung einer Betriebsversion

- Subversion [3]
- Apache Ant [4]

2.3. Installation

2.3.1. Quellcode beziehen

```
svn co https://www.europeanalabs.eu/svn/contrib/sti/branches/sti-gwt
```

oder für den DARIAH GeoBrowswer

```
svn co http://dev.dariah.eu/svn/repos/eu.dariah.de/ap1/sti-gwt-dariah-geobrowser
```

(TODO Prüfen, welche Funktionalitäten in die original Codebasis übernommen werden können):

- KMZ-Dateien laden (bereits in die Original Codebasis des Telplus Proxys übernommen)
- CSV-Dateien laden (JavaScript-Erweiterung im e4D)

2.3.2. Servlet kompilieren

Das Ant Script (build.xml) kompiliert per default eine installationsfähige Version.

```
ant
```

Ein WAR erhält man per:

```
ant war
```

2.3.3. Server-Adresse anpassen

Mit dem folgenden Kommando kann die Serverdresse angepasst werden. Die Serveradresse wird benötigt um anzugeben, wo sich die Kartenlayer, die Demodatensätze und der XHR Proxy befindet.

```
sed -i "s/139\.18\.13\.172/134\.76\.21\.92/g" *.*
```

Da inzwischen nur relative Pfade in den Konfigurationsdateien angegeben werden, funktioniert

```
sed -i "s/\/geoserver\/wms139/134\.76\.21\.92\/geoserver\/wns/g" *.*
```

vielleicht besser (bei mir und meinem Apple klappt beides nicht, ich will aber auch sowieso einen eigenen Geo-Server installieren, dann ist das nicht nötig, meine ich).

Weitere Anpassungen

- Die erzeugten Dateien werden im Verzeichnis **build** abgelegt. Wenn das Target **war** benutzt wurde, findet sich im Wurzelverzeichnis die Datei e4d.war.
- Im Verzeichnis **scripts/sti/properties.js** können die Farben (der Umkreisungen?) angepasst werde.
- Die Kartenlayer werden in der **layers.xml**,
- die Datenquellen in **datasources.json** konfiguriert.
- Der Gradient wird in **scripts/sti/STIMap.js** eingestellt (ist noch hartkodiert!), siehe Methode **setCanvas**, Zeile 683ff.
- Um diverse andere Kartenmaterialien einzubinden (OpenStreetMap, Google, Bing), können in der Datei **/scripts/sti/STIProps.js** die entsprechenden Boolean-Werte auf true/false gesetzt werden.
- Alle Anpassungen wurden im DARIAH SVN bereits angepasst, für einen DARIAH GeoBrowser muss nur der GeoServer auf dem selben Rechner installiert werden, sowie der Telplus Proxy (der für DARIAH ebenfalls noch ins SVN wandert).

2.3.4. Deployment

Entweder wird das komplette ge"ant"ete Verzeichnis **build/** in des Tomcat's webapps-Verzeichnis geworfen, oder aber die ge"ant"ete WAR-Datei **e4d.war**.

Anpassung des Google Maps API Keys

In der Datei **Sti.html** muss der Google Maps API Key eingetragen werden. Erzeugt werden kann ein solcher HIER.

2.4. Konfiguration

Der Loglevel wird in Datei **/webapps/e4d/WEB-INF/logging.properties** gesetzt.

2.5. Optionale Features

2.5.1. GeoServer

Der GeoServer [5] ist notwendig um eigenes Kartenmaterial zu verwenden.

Installation

Der Geoserver ist als WAR Datei verfügbar und kann direkt so in Tomcat deployed werden. Die Integration des eignen Kartenmaterials ist unter [6] beschrieben. Direkter Link zum Shapefile-Import siehe [7].

Wichtig dabei ist, dass

- die layers.xml editiert wird (also neues Kartenmaterial hinzugefügt wird per Serverangabe). Beispiele sind hier bereits vorhanden, und
- die korrekte EPSG angegeben wird beim Anlegen der neuen Layers, die Karten von [8] sind alle in EPSG:4326 kodiert, einige Karten hieraus sind in der Layers.xml bereits konfiguriert, nur noch der Server kann geändert werden, falls kein lokaler GeoServer installiert wurde. Stimmt die EPSG nicht, werden die Ausschnitte (BBOX) falsch berechnet und es wird Kwark angezeigt. Nach Eingabe der korrekten EPSG muss noch auf "Compute from native bounds" bei Punkt "Bounding Boxes" geklickt werden. Weitere Angaben sind beim Anlegen eines neuen Layers nicht zu machen.

Das Passwort für die GeoServer-Konfiguration (http://dev2.dariah.eu/geoserver/) bitte in Datei */webapps/geoserver/data/security/users.properties* suchen.

Konfiguration

Der GeoServer kann am besten über seine Web-Schnittstelle /geoserver/web/ administriert werden, auch das Logging und ähnliche Dinge.

2.5.2. XHR Proxy

Der XHR Proxy ist notwendig um KML von beliebigen URL in das Interface zu laden. Er umgeht die "same orgin" Regel in JavaScript.

Installation

Für die Installation sind zusätzlich noch die Werkzeuge Maven [9] und "patch" [10] notwendig.

```
svn co https://www.europeanalabs.eu/svn/contrib/kb/telplus
```

Vor dem Kompilieren werden die Patches angewandt, um einige Konfigurations-Dinge anzupassen (und um KMZ-Dateien einlesen zu können):

```
cd telplus
 patch -p0 < patches/sti.diff
 patch -p0 < patches/authorisation.patch
 patch -p0 < patches/dariah-kmz.patch
```

Mit

```
mvn package
```

wird abschließend kompiliert. Das im Verzeichniss **tpp/target/** generierte WAR-File kann ebenfalls in den Tomcat geworfen werden. Im README-File steht eine Kurzanleitung für die neue KMZ-Funktion.

In e4d muss nun nur noch ein neues Feld in die **datasources.json** eingefügt werden:

```
{
 "label" : "KMZ File URL",
 "url" : "/tpp/req?kmz=true&url="
 },
```

Fertig!

Konfiguration

Die Logdatei von tpp kann bei Bedarf angepasst werden in **/webapps/tpp-1.0.0-Beta10/WEB-INF/classes/log4j.xml**, oder die Logdatei**tpp.log** im Verzeichnis **/webapps/e4d/** die benötigten Schreibrechte geben, falls irgend etwas nicht schreibbar sein sollte...

Weiterhin muss darauf geachtet werden, dass entweder das telplus WAR **tpp.war** heißt, oder aber der Pfad zum telplus in**webapps/e4d/datasources.json** angepasst wird (unter "KML File URL").

Testen

Der Proxy kann mit einem direkten Aufruf getestet werden.
Z.B. http://134.76.21.92:8080/tpp/req?url= oder http://134.76.21.92:8080/tpp/req?url=http://www.spiegel.de/ (Im Browser aufrufen, sonst muss die hintere URL kodiert werden).

2.6. Beispiele/Testdateien

- KML
  - http://134.76.21.92:8080/exist/rest/db/archaeo18/queries/lab.xq
  - http://dev2.dariah.eu/locations_demo.kml
  - http://www.cceh.uni-koeln.de/projekte/tb/totenbuch.kml
  - http://steinheim-institut.de/daten/epidat.kml oder (eingebaut in eigenen Anwendung) http://steinheim-institut.de/cgi-bin/epidat?info=e4d

- KMZ
  - http://dev2.dariah.eu/locations_demo.kmz

2.7. Nice To Have

- **e4d could handle KMZ files as well**
  - it can now: see tpp configuration, single KML file in KMZ file only at the moment
- more ideas
  - GZIP-Dekoprimierungen in JavaScript
  - Einbau Client, bevorzugt letzteres
  - Tipps gibt's unter [11] und/oder [12]

2.8. Fehler und Bugfixes

1. Ohne lokale Kartenserver (keine Einträge in **layers.xml**) wird das linke Menu nicht angezeigt.
   - in **/war/scripts/sti/STIProps.js** den Wert für **historic maps** auf **false** setzen!

2.9. Quelle dieser Doku...

...https://wiki.sub.uni-goettingen.de/ent/index.php/Installation_E4D#GeoServer

TODO: Irgendwie so basteln, dass nicht zwei Dokumente gepflegt werden müssen!

2.10. Hinweis

Im DARIAH WIKI wird der GeoBrowser auf verschiedenen Seiten gepflegt, das ist nicht sehr sinnvoll, die geladene Seite und diese sollten zusammengeführt werden: https://dev2.dariah.eu/wiki/x/Awgj

3. Anhang 2 – GeoBrowser Betriebs-Dokumentation (Stand 22.04.2013)

3.1. GeoBrowser

GeoBrowser browses geographical data...

3.2. Basic Data

host	http://dev2.daria.eu/
responsible person	*Stefan E. Funk (SUB)*
installed software	Please see Technical Documentation Site: GeoBrowser (e4D)
availability	Just go to http://dev2.dariah.eu/e4d/

further information

- Please see Technical Documentation Site: GeoBrowser (e4D)

3.3. Installation Information

3.3.1. Binaries

- *e4d and the GeoServer are installed as Servlet WARs in the Tomcat's webapps folder: /usr/share/tomcat6/webapps/e4d.war and /usr/share/tomcat6/webapps/geoserver*.war

3.3.2. Sources

- Source code is available in the DARIAH SVN repository: http://dev.dariah.eu/svn/repos/eu.dariah.de/ap1/sti-gwt-dariah-geobrowser

- All changes for the DARIAH GeoBrowser are made there and are ready to compile, please see GeoBrowser (e4D) site

3.3.3. Config

- Please see Tomcat's webapp folder as mentioned above, see GeoBrowser documentation, and GeoServer configuration site: http://dev2.dariah.eu/geoserver/

3.3.4. User Accounts

- not needed for e4D, GeoServer admin credentials for configuration from browser please see the file */webapps/geoserver/data/security/users.properties*

3.3.5. Logging

- please see tomcat's log files and logging for
  - GeoServer: webapps/geoserver/data/logs/geoserver.log
  - e4D: Tomcat's catalina.out

3.3.6. Data

- Map material please see e4D documentation

3.3.7. Backup

The following files and directories should be backuped:

path policy

?? ??

?? ??

3.4. Configuration

3.4.1. Service Internal Settings

- *some Apache vhosts set für Port 80 and 443: please see there!*

3.4.2. Firewall Setup

- *nothing*

3.4.3. Monitoring

- *Monitored by JSC (Nagios)*