IIIF development at the BSB
Agenda

- IIIF for the current BSB Newspaper project
- IIIF products/components developed at the BSB
IIIF for the current BSB Newspaper project “digiPress”

➢ We are using self-implemented IIIF-Java-Libraries in Newspaper Portal-Webapp (project independent Java libraries for IIIF Image API 2.0 and IIIF Presentation API 2.0)
➢ pregenerated IIIF-manifest.json files, adding metadata section dynamically (to reflect metadata changes immediately)
➢ viewer: Mirador 2.0.0 (custom features: navigation icons, search word highlighting)
IIIIF for the BSB Newspaper project “digiPress”: Mirador viewer

Viewer “Mirador” showing a newspaper issue. (Navigation on the left hand side is not part of Mirador.)
IIIF for the BSB Newspaper project “digiPress”: Thumbnails

Thumbnail gallery of title pages of newspapers of 21. April 1929 using IIIF Image API to get 400px wide thumbnails: .../iiif/image/<identifier>/full/,400/0/native.jpg
IIIF for the BSB Newspaper project “digiPress”:
Thumbnail gallery

Thumbnail gallery of all pages of “Der gerade Weg” of 21. April 1929

1) getting Manifest as Java model of IIIF Presentation API library
2) iterating of default sequence and all contained canvases
3) using IIIF Image API to get 400px wide thumbnails
IIIF for the BSB Newspaper project “digiPress”:
Search results

Search result list using snippets

1) getting snippets over IIIF Image API
2) highlighting done with CSS
IIIF products/components developed at the BSB

- IIIF Image API 2.0 Java library
- IIIF Presentation API 2.0 Java library
- IIIF Server “Jiiif”
IIIF Image API 2.0 library

Features
- IIIF Image API 2.0 Level 2 “desirable” compliant
- Embeddable Spring components: Spring MVC Controller, Spring Service
- Access to images over customizable identifier/path resolving mechanism
- Java 8 Image I/O API based
- Supported formats: JPEG, PNG, (W)BMP, GIF, TIFF

Example URI:
IIIF Presentation API 2.0 library

Features

- Java model of IIIF Presentation API
- Access to manifest-files over customizable identifier/path resolving mechanism
- Security hook: customizable access restriction for each identifier (e.g. whitelist)
- Simple Manifest generator using directory with images of object
- Java 8 based
IIIF Server „Jiiif“
pronounce it "tʃiːf", like "chief"

Features

- On the fly image processing: No additional pregenerated (pyramid zoom) images are needed. No additional storage consumption.
- Standalone mode: Ready to deploy standard java web application (WAR) for running as a standalone IIIF server.
- Embedded mode: Embeddable Spring components: Spring MVC Controller, Spring Services for embedding IIIF functionality into existing Spring web applications
- Spring based modular, extendable, easy to maintain enterprise architecture
- Highly customizable image storage and identifier resolving: Access to images over project specific Resolver-plugin mechanism.
- Support for Filesystem- and HTTP-Image-Repositories (own protocols can be added by providing specific resolver)
- Highly customizable manifest generation: implement your own mapping from project specific structure metadata to standard Manifest object
Thank you