RFC 225 - Web Resources

Draft

11 Pages

Text in Red is here to help you. Delete it when you have followed the instructions.
The <RFC Title> can be set from the File>Properties:User Defined menu. To update it onscreen, press F9. To update all of the fields in the document Select All (CTRL-A), then hit F9. Set the release level by selecting one from: Draft, Final Draft, Release. The date is set automatically when the document is saved.

Abstract

10 point Arial Centered.

Web Applications require more and more access to web resources like Javascript files. These web resources can be served from content delivery network over the Internet but this has implications for HTTP/HTTPS access, security and availability (especially in the future). Therefore, applications serve these web resources from their own server. The OSGi environment is a perfect environment to handle these web resources. This RFC defines how to package Web Resources in an OSGi system.

0 Document Information

0.1 License

DISTRIBUTION AND FEEDBACK LICENSE, Version 2.0

The OSGi Alliance hereby grants you a limited copyright license to copy and display this document (the “Distribution”) in any medium without fee or royalty. This Distribution license is exclusively for the purpose of reviewing and providing feedback to the OSGi Alliance. You agree not to modify the Distribution in any way and further agree to not participate in any way in the making of derivative works thereof, other than as a necessary result of reviewing and providing feedback to the Distribution. You also agree to cause this notice, along with the accompanying consent, to be included on all copies (or portions thereof) of the Distribution. The OSGi Alliance also grants you a perpetual, non-exclusive, worldwide, fully paid-up, royalty free, limited license (without the right to sublicense) under any applicable copyrights, to create and/or distribute an implementation of the Distribution that: (i) fully implements the Distribution including all its required interfaces and functionality; (ii) does not modify, subset, superset or otherwise extend the OSGi Name Space, or include any public or protected packages, classes, Java interfaces, fields or methods within the OSGi Name Space other than those required and authorized by the Distribution. An implementation that does not satisfy limitations (i)-(ii) is not considered an implementation of the Distribution, does not receive the benefits of this license, and must not be described as an implementation of the Distribution. "OSGi Name Space" shall mean the public class or interface declarations whose names begin with "org.osgi" or any recognized successors or replacements thereof. The OSGi Alliance expressly reserves all rights not granted pursuant to these limited copyright licenses including termination of the license at will at any time.

EXCEPT FOR THE LIMITED COPYRIGHT LICENSES GRANTED ABOVE, THE OSGi ALLIANCE DOES NOT GRANT, EITHER EXPRESSLY OR IMPLIEDLY, A LICENSE TO ANY INTELLECTUAL PROPERTY IT, OR ANY THIRD PARTIES, OWN OR CONTROL. Title to the copyright in the Distribution will at all times remain with the OSGi Alliance. The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted therein are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

THE DISTRIBUTION IS PROVIDED "AS IS," AND THE OSGi ALLIANCE (INCLUDING ANY THIRD PARTIES THAT HAVE CONTRIBUTED TO THE DISTRIBUTION) MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE CONTENTS OF THE DISTRIBUTION ARE SUITABLE FOR ANY PURPOSE; NOR THAT THE IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.
NEITHER THE OSGi ALLIANCE NOR ANY THIRD PARTY WILL BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY USE OR DISTRIBUTION OF THE DISTRIBUTION.

Implementation of certain elements of this Distribution may be subject to third party intellectual property rights, including without limitation, patent rights (such a third party may or may not be a member of the OSGi Alliance). The OSGi Alliance is not responsible and shall not be held responsible in any manner for identifying or failing to identify any or all such third party intellectual property rights.

The Distribution is a draft. As a result, the final product may change substantially by the time of final publication, and you are cautioned against relying on the content of this Distribution. You are encouraged to update any implementation of the Distribution if and when such Distribution becomes a final specification.

The OSGi Alliance is willing to receive input, suggestions and other feedback (“Feedback”) on the Distribution. By providing such Feedback to the OSGi Alliance, you grant to the OSGi Alliance and all its Members a non-exclusive, non-transferable,

worldwide, perpetual, irrevocable, royalty-free copyright license to copy, publish, license, modify, sublicense or otherwise distribute and exploit your Feedback for any purpose. Likewise, if incorporation of your Feedback would cause an implementation of the Distribution, including as it may be modified, amended, or published at any point in the future ("Future Specification"), to necessarily infringe a patent or patent application that you own or control, you hereby commit to grant to all implementers of such Distribution or Future Specification an irrevocable, worldwide, sublicenseable, royalty free license under such patent or patent application to make, have made, use, sell, offer for sale, import and export products or services that implement such Distribution or Future Specification. You warrant that (a) to the best of your knowledge you have the right to provide this Feedback, and if you are providing Feedback on behalf of a company, you have the rights to provide Feedback on behalf of your company; (b) the Feedback is not confidential to you and does not violate the copyright or trade secret interests of another; and (c) to the best of your knowledge, use of the Feedback would not cause an implementation of the Distribution or a Future Specification to necessarily infringe any third-party patent or patent application known to you. You also acknowledge that the OSGi Alliance is not required to incorporate your Feedback into any version of the Distribution or a Future Specification.

I HEREBY ACKNOWLEDGE AND AGREE TO THE TERMS AND CONDITIONS DELINEATED ABOVE.

0.2 Trademarks

OSGi™ is a trademark, registered trademark, or service mark of the OSGi Alliance in the US and other countries. Java is a trademark, registered trademark, or service mark of Oracle Corporation in the US and other countries. All other trademarks, registered trademarks, or service marks used in this document are the property of their respective owners and are hereby recognized.

0.3 Feedback

This document can be downloaded from the OSGi Alliance design repository at https://github.com/osgi/design The public can provide feedback about this document by opening a bug at https://www.osgi.org/bugzilla/.

0.4 Table of Contents

0.5 Terminology and Document Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY" and "OPTIONAL" in this document are to be interpreted as described in 10.1.

```
Source code is shown in this typeface.
```

0.6 Revision History

The last named individual in this history is currently responsible for this document.

Revision	Date	Comments
Initial	May 2016	David Bosschaert, convert RFP into RFC.

1 Introduction

Web Applications require more and more access to web resources like Javascript files. These web resources can be served from content delivery network over the Internet but this has implications for HTTP/HTTPS access, security and availability. Therefore, applications often serve these web resources from their own server. The OSGi environment is a perfect environment to handle these web resources since it has a very clear life cycle. This RFC defines how to package Web Resources in an OSGi system.

This RFC originates from the OSGi enRoute work. In this project, a number of services were identified, designed and implemented based on their needs for web based applications. This document analyzes the application domain and defines the problem that need to be solved.

2 Application Domain

A w*eb resource* is a an artifact that must be available from an HTTP(S) server so that other web resources and/or dynamic content can refer to it via a URL. A web resource is static, it does not require any processing before it is delivered. For example, the Angular JS Javascript library contains of a coupled of hundred files that need to be available to the HTML page that contains the application.

2.1 CDNs

The preferred mode to serve static resources is over a Content Delivery Network. A CDN provides high bandwidth and since the same URL is shared between many applications the browser has a higher chance of finding the web resource in its cache.

However, in practice CDN are not always a correct solution.

- Many applications have a governing process that requires that any web resources are locally served so their content in runtime can be verified.
- HTTPS served pages require their resources to be served over HTTPS as well, which are not always available in the CDN.
- Though many CDNs will be around forever, it is not clear that over long term all the content will be available, potentially killing an application in the future.
- A CDN requires network connectivity to the Internet. Malfunctions and closed intranets make them sometimes an impossibility.

For these reasons, *web applications* frequently need to serve web resources from their own servers. Since these resources do not have to be dynamically generated they are usually served separately by web servers that are highly optimized and provide extensive caching support for static resources.

However, in Java it is also popular to deliver web resources from the JAR files. The WAR format specifies that any content in the `META-INF/resource` directory in a JAR in the WAR files `lib` directory is served as a web resource, the remaining part of the path of the resource is then mapped to the root of the web server. For example, the web resource `META-INF/resource/angular/1.3.1/angular.js` will be available as `/angular/1.3.1/angular.js`. This model is described in Error: Reference source not found

2.2 Paths

The paths to the web resources are generally unmanaged. The URLs to the web resources are used in many different places in a typical project. Managing these paths can be quite a challenge. These paths are also shared between other applications that run on the same web server. To prevent, the often short names, of the web resources to clash with other applications, some application servers can rewrite URLs so that each application gets its own namespace.

A simple solution would be to use long path names but this is not user friendly, there is a desire to have short URLs.

2.3 Caching

The HTTP(S) protocol has extensive support for caching. Headers in the protocol provide information to proxies and the browser of what can and what can not be cached and how long. Over time a large number of best practices have been evolved that are non-trivial to manage.

Caching is crucial for good performance since javascript applications have been growing up over the last few years. Downloading a megabyte in compressed Javascript is no exception anymore. The average page in 2014 includes about 300k of Javascript across 18 different files (HTTP Archive Error: Reference source not found) This number is likely to increase.

Debugging must take caching into account. With a development IDE like bndtools the edit-build-debug cycle is very short that it becomes fully interactive. However, caches can slow this down if they need to be cleared for each cycle.

2.4 Merging

A solution that is becoming more popular is merging all the Javascript sources in a single resource. This is much more efficient than the dependency manager in the browser going back and forth and traversing the transitive dependency tree. Merging can then also include Javascript files from *application plugins*. Since these plugins are then loaded they get control and register themselves with the application or a module system like Angular.

During the debug cycle, there is often also the desire to use Javascript files that are not *minified*.

2.5 Dependencies

Most Javascript libraries use `package.json` files defined in Error: Reference source not found These files are similar to pom.xml in maven. They describe the library and the names of the their dependencies. There are alternatives with bower Error: Reference source not found and others. Most dependency managers work by searching a number of directories defined by a path.

Through these dependency managers it is possible to find the closure of Javascript web resources that should work together.

2.6 Wrapping

Wrapping web resources in a bundle creates a dependency on the development life cycle of the web resources. For example, if a new Angular JS is released then developers need to quickly have access to this web resource. However, in the WAR model a new lib JAR must be created with the content. Since each developer that uses Angular has this problem a group effort would be beneficial. This is the same problem as OSGi has with bundle wrapping. In this case, a group created webjars.org Error: Reference source not found In this project, they allow people to define a Github project that pushes a JAR to maven central that stores the web resources in `META-`

INF/resource/<name>/<version>. The webjars.org site then creates an index on maven central. This project is quite popular and contains hundreds of webjars.

In the webjar.org model dependencies are handled through maven.

2.7 OSGi enRoute

In OSGi enRoute a “web resource server” servlet was developed to handle web resources. Initially the `static` directory in the bundle was mapped to the server but now also the `META-INF/resource` directory is supported. The web resource server would then handle searching and overlapping directories.

Since OSGi had the requirement-capability model, OSGi enRoute also developed a `osgi.enroute.webresource` namespace. This namespace defined the name of the capability to be the web resource path. Bundles providing this path would then add a capability with this name:

```
Provide-Capability: osgi.enroute.webresource; \

  osgi.enroute.webresource='/google/angular/1.3'; \

  version='1.3.1'
```

Using the Manifest Annotations in bnd, it was then possible to require a version of Angular in the application code.

2.8 Terminology + Abbreviations

3 Problem Description

Managing web resources is an important aspect of any web application. Handling these web resources developers encounter the following problems:

- Managing the dependencies of the Javascript resources is error prone and hard to do. It is not possible to use the OSGi Require-Capability model to create a closure of web resources
- There is no way to map OSGi bundled web resources to a web path
- Managing the web paths for the resources is cumbersome, especially if each resource needs its own versioned path
- Difficult to manage plugins of an application that require additional web resources.
- There is no easy way to use the results of the webjars.org project

Therefore, this RFP seeks a specification that defines how web resources can be stored in bundles while taking advantage of the Require-Capability model.

4 Requirements

4.1 General

- G0010 – Provide a mechanism to wrap web resources as bundles
- G0020 – Define how web resources are mapped to the web path
- G0030 – Define how web resource bundles can be used with the Require-Capability model.
- G0040 – Allow the use of Javascript dependency managers like npm and/or bower to control the Requirements and Capabilities for automatic creation of wrapped bundles.
- G0050 – Provide a web path scheme to make sure names do not clash

4.2 Merged Resources

- M0010 – Provide a way to have a bundle dependent URL that merges all javascript resources needed for an HTML page
- M0020 – Allow priorities, e.g. Angular needs before any angular modules
- M0030 – Allow application plugins to be delivered as bundles.
- M0040 – Define a debug mode where web resources are not cached
- M0050 – Define a debug mode where it is possible to switch the minified sources for the original sources, if they are available.

5 Technical Solution

First give an architectural overview of the solution so the reader is gently introduced in the solution (Javadoc is not considered gently). What are the different modules? How do the modules relate? How do they interact? Where do they come from? This section should contain a class diagram. Then describe the different modules in

detail. This should contain descriptions, Java code, UML class diagrams, state diagrams and interaction diagrams. This section should be sufficient to implement the solution assuming a skilled person.

Strictly use the terminology a defined in the Problem Context.

On each level, list the limitations of the solutions and any rationales for design decisions. Almost every decision is a trade off so explain what those trade offs are and why a specific trade off is made.

Address what security mechanisms are implemented and how they should be used.

6 Data Transfer Objects

RFC 185 defines Data Transfer Objects as a generic means for management solutions to interact with runtime entities in an OSGi Framework. DTOs provides a common, easily serializable representation of the technology.

For all new functionality added to the OSGi Framework the question should be asked: would this feature benefit from a DTO? The expectation is that in most cases it would.

The DTOs for the design in this RFC should be described here and if there are no DTOs being defined an explanation should be given explaining why this is not applicable in this case.

This section is optional and could also be provided in a separate RFC.

7 Javadoc

Please include Javadoc of any new APIs here, once the design has matured. Instructions on how to export Javadoc for inclusion in the RFC can be found here: https://www.osgi.org/members/RFC/Javadoc

8 Considered Alternatives

For posterity, record the design alternatives that were considered but rejected along with the reason for rejection. This is especially important for external/earlier solutions that were deemed not applicable.

9 Security Considerations

Description of all known vulnerabilities this may either introduce or address as well as scenarios of how the weaknesses could be circumvented.

10 Document Support

10.1 References

[1]. Bradner, S., Key words for use in RFCs to Indicate Requirement Levels, RFC2119, March 1997.

[2]. Software Requirements & Specifications. Michael Jackson. ISBN 0-201-87712-0

Add references simply by adding new items. You can then cross-refer to them by chosing <Insert><Cross Reference><Numbered Item> and then selecting the paragraph. ***STATIC REFERENCES (I.E. BODGED) ARE NOT ACCEPTABLE, SOMEONE WILL HAVE TO UPDATE THEM LATER, SO DO IT PROPERLY NOW.***

10.2 Author's Address

Name	Peter Kriens
Company	aQute
Address	9c, Avenue St. Drezery
Voice	+33 467542167
e-mail	Peter.kriens@aQute.biz

Name	David Bosschaert
Company	Adobe Systems
Address	
Voice	
e-mail	bosschae@adobe.com

10.3 Acronyms and Abbreviations

10.4 End of Document