

Site aware exception handling in Magnolia CMS

- [Introduction](#)
- [Set up steps](#)
 - [Setting Servlet Container exception handling page](#)
 - [Deploy Java implementation of site aware exception handling](#)
 - [Sample Java code](#)
 - [Deployment for light-dev and on Magnolia Cloud](#)
 - [Configure mappings for your error locations](#)
 - [YAML configuration](#)
 - [JCR configuration](#)
 - [Creating different exception pages](#)
 - [Changing exception page location](#)
 - [Changing exception page template](#)
 - [Sample FTL code](#)
- [References](#)

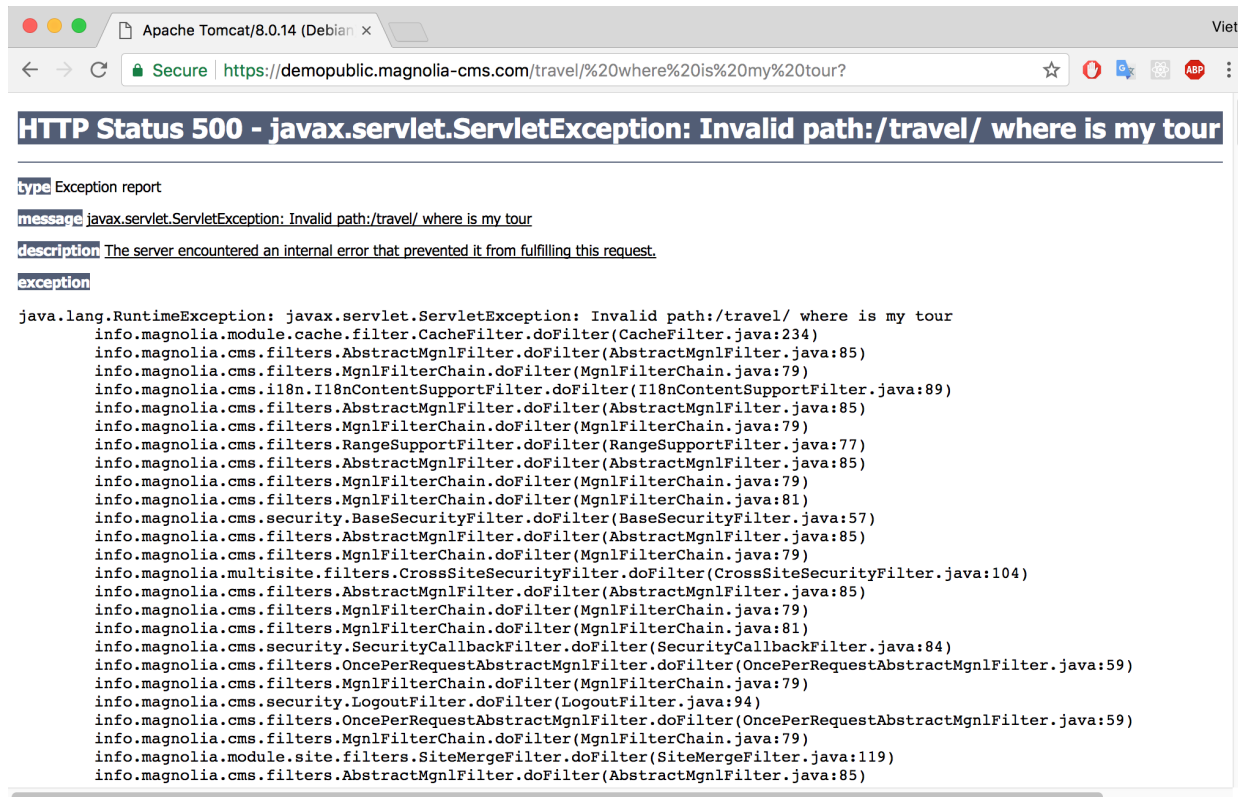
 This page has been updated to comply with Magnolia 5.7.+ as of February 2019, previous version was written for Magnolia older versions.

Introduction

There have been more and more multi-site users of Magnolia CMS asking us for a customizable, multi-language enabled, site aware and out of the box solution from Magnolia CMS to support web container servlet exception handling. As a multi-site user of Magnolia CMS, you can have multiple sites configured and mapped by multiple URLs / Domains / Hosts. Whenever an issue happens such as Resource not found (HTTP 404 status code) or server error (500 internal server error), web container will redirect us to its default error report page. If we don't have corresponding configuration, we cannot have a nice error report to our valued customers.

A typical example is when end user calling this link <https://demopublic.magnolia-cms.com/travel/ where is my tour?>

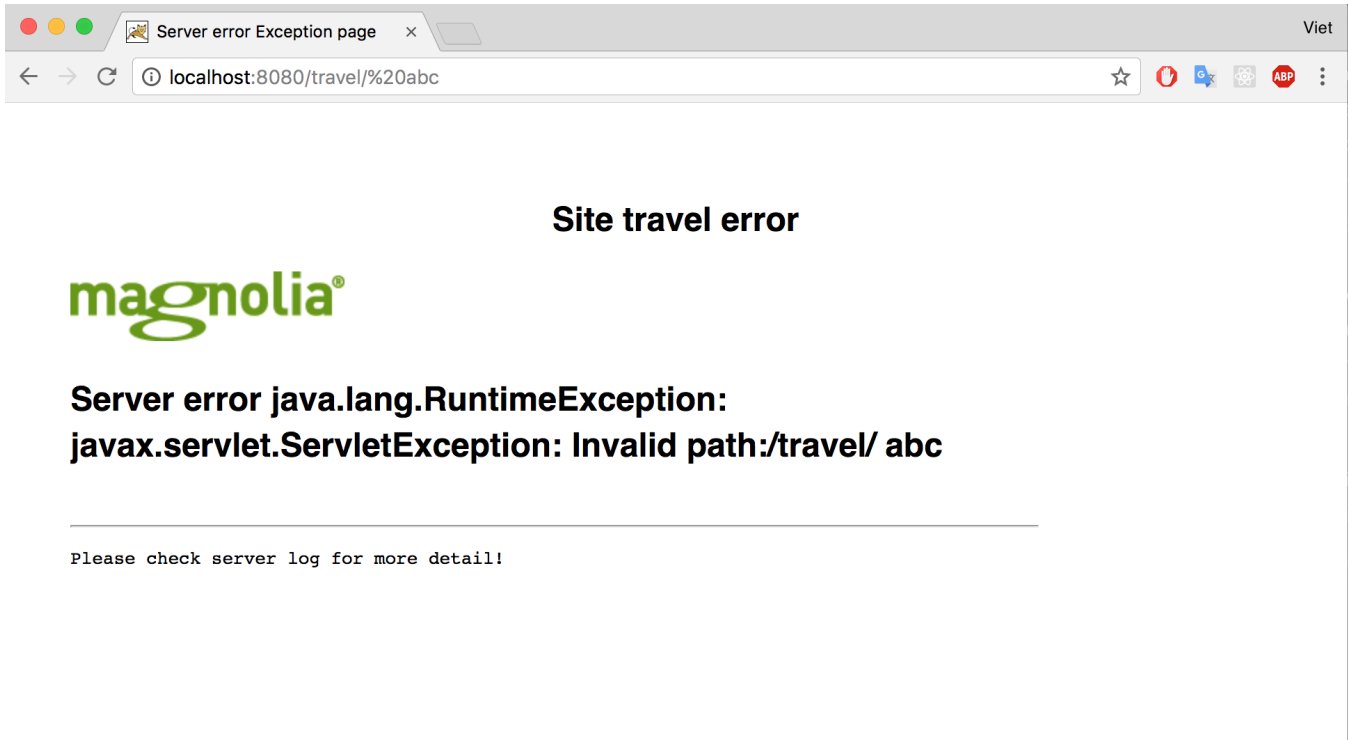
This is what we get from a non-configured site



```
java.lang.RuntimeException: javax.servlet.ServletException: Invalid path:/travel/ where is my tour
    info.magnolia.module.cache.filter.CacheFilter.doFilter(CacheFilter.java:234)
    info.magnolia.cms.filters.AbstractMgnlFilter.doFilter(AbstractMgnlFilter.java:85)
    info.magnolia.cms.filters.MgnlFilterChain.doFilter(MgnlFilterChain.java:79)
    info.magnolia.cms.i18n.I18nContentSupportFilter.doFilter(I18nContentSupportFilter.java:89)
    info.magnolia.cms.filters.AbstractMgnlFilter.doFilter(AbstractMgnlFilter.java:85)
    info.magnolia.cms.filters.MgnlFilterChain.doFilter(MgnlFilterChain.java:79)
    info.magnolia.cms.filters.RangeSupportFilter.doFilter(RangeSupportFilter.java:77)
    info.magnolia.cms.filters.AbstractMgnlFilter.doFilter(AbstractMgnlFilter.java:85)
    info.magnolia.cms.filters.MgnlFilterChain.doFilter(MgnlFilterChain.java:79)
    info.magnolia.cms.filters.MgnlFilterChain.doFilter(MgnlFilterChain.java:81)
    info.magnolia.cms.security.BaseSecurityFilter.doFilter(BaseSecurityFilter.java:57)
    info.magnolia.cms.filters.AbstractMgnlFilter.doFilter(AbstractMgnlFilter.java:85)
    info.magnolia.cms.filters.MgnlFilterChain.doFilter(MgnlFilterChain.java:79)
    info.magnolia.multisite.filters.CrossSiteSecurityFilter.doFilter(CrossSiteSecurityFilter.java:104)
    info.magnolia.cms.filters.AbstractMgnlFilter.doFilter(AbstractMgnlFilter.java:85)
    info.magnolia.cms.filters.MgnlFilterChain.doFilter(MgnlFilterChain.java:79)
    info.magnolia.cms.filters.MgnlFilterChain.doFilter(MgnlFilterChain.java:81)
    info.magnolia.cms.security.SecurityCallbackFilter.doFilter(SecurityCallbackFilter.java:84)
    info.magnolia.cms.filters.OncePerRequestAbstractMgnlFilter.doFilter(OncePerRequestAbstractMgnlFilter.java:59)
    info.magnolia.cms.filters.MgnlFilterChain.doFilter(MgnlFilterChain.java:79)
    info.magnolia.cms.security.LogoutFilter.doFilter(LogoutFilter.java:94)
    info.magnolia.cms.filters.OncePerRequestAbstractMgnlFilter.doFilter(OncePerRequestAbstractMgnlFilter.java:59)
    info.magnolia.cms.filters.MgnlFilterChain.doFilter(MgnlFilterChain.java:79)
    info.magnolia.module.site.filters.SiteMergeFilter.doFilter(SiteMergeFilter.java:119)
    info.magnolia.cms.filters.AbstractMgnlFilter.doFilter(AbstractMgnlFilter.java:85)
```

This guideline provide you with how to have it and how it has been made so that you can easily use it and customize it based on your practical situation.

After successfully installed the module, when accessing a nonexistence page such as <http://localhost:8080/travel/ abc>, you will have this:



Set up steps

Setting Servlet Container exception handling page

First of all in order to make our web-container such as Apache Tomcat aware of our customization, we would have to put snippet of code this to our "*Tomcat* [at/webapps/YOUR-MAGNOLIA-WEBAPP/WEB-INF/web.xml](#)" file just above the closing of 'web-app' tag (above this "</web-app>").

```
<error-page>
  <error-code>404</error-code>
  <location>/customException</location>
</error-page>
<error-page>
  <error-code>500</error-code>
  <location>/customException</location>
</error-page>
<error-page>
  <exception-type>java.lang.Throwable</exception-type>
  <location>/customException</location>
</error-page>
```

Please note that we are using a specific location "/customException" for all every "java.lang.Throwable" ones, you could also change it in case of conflict with any of your existing one. Later on we will show where the mapping is. So if you change it here, you would also have to change the corresponding one for direct mapping of exception handling page.

Deploy Java implementation of site aware exception handling

If you are developing a custom Magnolia Maven module, then just make below class available in your classpath.

Sample Java code

Class: *info.magnolia.virtualuri.mapping.SiteAwareExceptionMapping*

SiteAwareExceptionMapping

```
package info.magnolia.virtualuri.mapping;

import java.net.URI;
import java.util.Optional;

import javax.servlet.http.HttpServletRequest;

import info.magnolia.context.MgnlContext;
import info.magnolia.module.site.SiteManager;
import info.magnolia.objectfactory.Components;

public class SiteAwareExceptionMapping extends DefaultVirtualUriMapping {

    public static final String SERVLET_ERROR_REQUEST_URI_ATTRIBUTE_NAME = "javax.servlet.error.request_uri";

    private String siteName;

    @Override
    public Optional<Result> mapUri(URI uri) {

        HttpServletRequest request = MgnlContext.getWebContext().getRequest();
        Object errorRequestObject = request.getAttribute(SERVLET_ERROR_REQUEST_URI_ATTRIBUTE_NAME);
        if (errorRequestObject == null) {
            return Optional.empty();
        }

        String errorRequest = errorRequestObject.toString();
        String[] requestParts = errorRequest.split("/");

        if (requestParts.length >= 3) {
            errorRequest = "/" + requestParts[2] + "/"; // remove some parts due to Magnolia specific
implementation
        }

        String requestedSite = Components.getComponent(SiteManager.class).getAssignedSite(request.
getServerName(), errorRequest).getName();

        if (!siteName.equals(requestedSite)) {
            return Optional.empty();
        }

        return super.mapUri(uri);
    }

    public String getSiteName() {
        return siteName;
    }

    public void setSiteName(String siteName) {
        this.siteName = siteName;
    }
}
```

This source code [SiteAwareExceptionMapping.java](#) is provided without any guarantee as a community creative & publicly available one. Please use it with cares.

Deployment for light-dev and on Magnolia Cloud

Customers of Magnolia light-dev and Magnolia Cloud could also deploy a Java class into Magnolia using Groovy Classes follow similar to our guideline here [Model Class](#)

1._Create corresponding folder structure and a SiteAwareExceptionMapping file using [Magnolia Groovy App](#) as below image:

Name	Enabled	Status	Modification date
<input type="checkbox"/> deleteSomeContacts	false	○	Feb 13, 2018 2:34 PM
<input type="checkbox"/> importContactsFromXml	false	○	Feb 13, 2018 3:52 PM
<input type="checkbox"/> simplifiedDataHierarchyNavigation	false	○	Feb 13, 2018 3:03 PM
<input type="checkbox"/> createAppScript	false	○	Dec 14, 2017 10:08 PM
<input type="checkbox"/> samples	—	○	Feb 20, 2018 2:18 PM
<input type="checkbox"/> info	—	○	Feb 27, 2019 2:27 PM
<input type="checkbox"/> magnolia	—	○	Feb 27, 2019 2:27 PM
<input type="checkbox"/> virtualuri	—	○	Feb 27, 2019 2:27 PM
<input type="checkbox"/> mapping	—	○	Feb 27, 2019 2:31 PM
<input checked="" type="checkbox"/> SiteAwareExceptionMapping	true	○	Feb 27, 2019 5:06 PM

2. Edit the empty file and put above Java source code inside.

3. Be aware of this issue (and workaround) when registering new classes: [MGNLGROOVY-148](#) - Creating a new "groovy class" always fails with compilation error [OPEN](#)

(work around: click the "Is a script?" checkbox when editing the file, save. This allows the node to be created for the script. Edit the script and uncheck the box.)

→ Above steps make the class available in your class path without custom module or additional deployment efforts.

Configure mappings for your error locations

After previous step, you already configured Servlet container to render your exception in your specified page / servlet called "customException".

Please be careful that within your exception page, if you also having the same kind of exception, a cyclic reference will happen which might lead to a "stack overflow" error while rendering the exception of the exception page 🤔

Please use either YAML configuration or JCR configuration, do not use both of them to prevent duplication.

YAML configuration

Under your light module, create folder name 'virtualUriMappings' and put your configured mappings inside such as:

[travel-exception-mapping.yaml](#)

[sportstation-exception-mapping.yaml](#)

[fallback-exception-mapping.yaml](#)

Sample content for your convenience, please change the mapping detail based on your needs:

travel-exception-mapping.yaml

```
class: "info.magnolia.virtualuri.mapping.SiteAwareExceptionMapping"  
fromUri: "/customException"  
toUri: "redirect:/travel/exception"  
siteName: "travel"
```

JCR configuration

All the configuration points should be located under any of your module or light module such as 'mgnsupport' module in this case:

Node name	Value	Type	Status	Modification date
mgnsupport	—	—	○	Jan 24, 2018 3:09 PM
templates	—	—	○	Jan 24, 2018 3:09 PM
pages	—	—	○	Jan 24, 2018 3:09 PM
exception	—	—	○	—
renderType	freemarker	String	—	—
templateScript	/mgnsupport/templates/pages/exception.ftl	String	—	—
title	Exception handling template	String	—	—
virtualUriMappings	—	—	○	Jan 24, 2018 3:09 PM
travel-exception-mapping	—	—	○	Jan 24, 2018 3:09 PM
class	info.magnolia.virtualuri.mapping.SiteAwareExceptionMapping	String	—	—
fromUri	/.exception	String	—	—
siteName	travel	String	—	—
toUri	forward:/travel/exception	String	—	—
fallback-exception-mapping	—	—	○	Jan 24, 2018 3:09 PM
class	info.magnolia.virtualuri.mapping.SiteAwareExceptionMapping	String	—	—
fromUri	/.exception	String	—	—
siteName	fallback	String	—	—
toUri	forward:/exception	String	—	—
version	1.0.0-SNAPSHOT	String	—	—

Basically we provided a default exception page template, its configuration is under our module 'templates/pages/exception' node as any other template configuration. You can reference to our official documentation for template configuration guidelines.

Creating different exception pages

Because we need to provide 2 site-aware exception mappings (in this example) which have been configured out of the box pointing to 2 default exception pages for our 'travel' demo site and the 'fallback' one.

So you should create 2 default exception pages under our pages app as below image and assign them using your configured template(s).

localhost:8080/magnolia/admincentral#app:pages:browser;/exception:treeview:

magnolia

PAGES

Search

Page	Title	Template	Status	Modification date
<input type="checkbox"/> travel	Travel Home	Travel Home	●	Jan 24, 2018 3:07 PM
<input type="checkbox"/> tour-type	Tour Types	TourType Category Overview	●	Oct 27, 2015 4:41 PM
<input type="checkbox"/> destination	Destinations	Destination Category Overvi	●	Oct 27, 2015 7:14 PM
<input type="checkbox"/> tour	Tour detail	Tour Detail	●	Oct 27, 2015 7:36 PM
<input type="checkbox"/> stories	Stories	List of stories	●	Aug 10, 2017 11:31 PM
<input type="checkbox"/> about	About	Travel Standard	●	Oct 27, 2015 7:50 PM
<input type="checkbox"/> contact	Contact	Travel Standard	●	Oct 30, 2015 5:44 PM
<input type="checkbox"/> meta	Container for meta pages	Travel Standard	●	Jun 4, 2015 1:37 AM
<input type="checkbox"/> book-tour	Book your tour	Travel Standard	●	Dec 17, 2015 6:51 PM
<input type="checkbox"/> members	Members	Travel Standard	●	Nov 25, 2015 5:37 PM
<input checked="" type="checkbox"/> exception	Travel site exception	Exception handling template	○	Jan 24, 2018 3:02 PM
<input type="checkbox"/> sportstation	Sportstation	Travel Home	●	Jul 1, 2015 3:44 PM
<input checked="" type="checkbox"/> exception	exception	Exception handling template	○	Jan 18, 2018 5:15 PM

2 Items selected

i Your exception information follow Java Servlet Specification will be stored within current request attribute named "javax.servlet.error.exception".

Using Magnolia Freemarker renderer (FTL file) you can call `#{ctx.request.getAttribute("javax.servlet.error.exception")}` to get it. Some other error fields that you can get from Servlet API:

ServletRequest	javax.servlet.error.status_code	Integer	for error pages only: HTTP status code
ServletRequest	javax.servlet.error.exception_type	Class	for error pages only: exception class
ServletRequest	javax.servlet.error.message	String	for error pages only: error message
ServletRequest	javax.servlet.error.exception	Throwable	for error pages only: exception
ServletRequest	javax.servlet.error.request_uri	String	for error pages only: original request URI
ServletRequest	javax.servlet.error.servlet_name	String	for error pages only: servlet name

Changing exception page location

As you can easily find that we have this configuration point `'/modules/mgnlsupport/virtualUriMappings/travel-exception-mapping@toUri'` which has the value as `'redirect:/travel/exception'` then if you have another page somewhere for your site, you can easily change the page location from `'/travel/exception'` to your new one. The working mechanism is described in our official Virtual URI mapping function.

Changing exception page template

We already provided you with a sample (not very fancy) page template within our pre-built package under `'/mgnlsupport/src/main/resources/mgnlsupport/templates/pages/exception.ftl'` (in this case my module name is mgnlsupport, will change it later when I have time). Then we have a configuration point for it such as

```
'/modules/mgnlsupport/templates/pages/exception@templateScript=/mgnlsupport/templates/pages/exception.ftl'
```

Then just follow our light-dev guideline, provide your own template and point to to appropriate classpath folder then the new FTL file will be used for our configured template.

Sample FTL code

```

<!DOCTYPE html>
<html>

<head>
  <title>Server error Exception page</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
  <style type="text/css">
    div {
      font-family: sans-serif;
      line-height: 1.4;
    }
    #wrapper {
      width: 100%;
      text-align: center;
    }
    #boxer {
      display: inline-block;
      margin: 40px;
    }
  </style>
</head>

<body>
[#assign mySite=sitefn.site()]
<div id="wrapper">
  <div id="boxer">
    <h2>Site ${mySite.name} error</h2>
    <div style="width: 80%; text-align: left;">
      <div><img src='data:image/gif;base64,R0lGODlhxAyAMQQAgaZALLMf4yzQNNlv/X573CfEMXZn40sMKC
/YOLsz3mmIOzy35a5UM/fr6nGcLzSj
///wAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACH5BAEAAABAALAAAAADFADIAAAAX
/ICSOZGmeaKqubOu+sArM9BnQcxDvfO
//wGALN7MRdcKkcs1srogAIw7prFqvWBlRSqP2CIHCjLHImS9Z6MkgaLcNwEPBMTAoCgm0fp9UYw0AAzMKAwoCfIiJMX5XbRAADwUBD1GKSGsDDQ
GbnJwNAwSWLIxWAg6PM58AnQFwBAYMbnQ1DQ5upnkqCw+xtw4GoTELBggCUMc0BQKszJ2usG4IDS0JAbcMD8EnpCMDrAMn1b2ywCQMB48IAA2URA
IGYlAMoQkKyALaJATqyDMI+SksOLDXr6BBKAIaxCNYIFc4Y8gcABTBTcSNKSUWQOwnUQQgNgkYOCjA7+CMA+36
/x0AmGBhQTWq2CA740BTGQKZAhhwicwlfJgmABlcaaIihItDSLQ0SRSCogQBRiogwLMqgEMkNh6cemLazmMIHKLwWvABJ6sA0NH12XSEUaQ5SBBk
OsLATqX/UKEx9CGcEga8mH5iAi6MAOBclf+ba21cEaz8FEZR4e0Qp5H6Nix5jEIBBwQOdITMo8XeGgAcDMiU+WWJ1MrEsCMyVR
/HYMs+0RxAGQFLnsTJutwyubNm0gdTEEKaAogA4hkVEDI9YALg6GQOSSwjfQcJBv8MwdnPX+3pEvehyGwoTRERw8PElChsR0dI9iZTWtxEB7xFKZ
gjs4TDRRCLMFogIC/SDVf8MAR5TGw7
/NViEY1A4Z5E7JFCGkQkEKjehfjQowCEUExnI3wpaHegUZjw0UNCD+WUoHIAMmZAAESK+F6NuxLmQ4nI4LCgjfFntN1ZqSJKFw2HVEQGbc94hM5q
HlZTwoGszjFzjFRyEdeRSsq545CmgTgmBCme2ABuBx1WUlcqPPYdjB8WCR9+q5hZp4ZJaddkP0DSIKSodY6Qp19smnRYikvu4FqfXZpw5VE9ktA
kmYVeuKFjJL6IAoaaEWmokSikqmh/yGjpa44qGrUpPKdYCCmVZIQ66hWbVpmqGceSiNa4HU6w38pEGBqMv+9OiOe89FZK59fQrBdVbpepWf
/rbjScNhqAliIJqk7NMPat5mFEaq5SoLX5bX0uplnqUSqa6VXG4paLuSkpriRL600ICBOIB2HJIPOPDnDCSV6+wJk0qolli4ujvcprMVwHC9mA6
6cL6NfpuMrOB6dDBaeE2jhagiTEqVk
/GBunHLm7oss44ar0pvxynCNu0Mh1wbjIldSbAZqGLwFkDCuN7c4xNHuDCzpcQ4NOJlFJMhLesVluzutqKsFoBAw
/w6AALuJbXfVZxuVsBIh1sYYpzbALcrRCYqkDYj2qtTj1dS7uXsGakXaucPJ1y35xVQwr1QXrzmqnHfUMA8F7+pIEx5IVKqBJAhENxGN2d89S4
/729ggtD1Up8EDSQQgeFNE0tEXC4irS/Su1nwa5d7Y8KzWyA+IZpMDRT/7guo3CSgJnIZ5rCi19k68y6WDO2MDKGqyU640bnJXhTsfCFmQI8J
/4wAz2nQqkBsIlMPC95wAB/8mVEOGcferS3u9KDDsMjJPyoBKA4rHvwIa8AX/otw9agKnAzqQf9UInwLd4b4HWjARw2DAZWhiALGkxiwC+F8
/JHHBEp7hFcJiQAVZgIkHIECEAWugCwf4hZ
/haIU8IEADEAC7XdHwh0BQyDEKYJ8kgMEgBASiEldaOwUkEQing0Kz1khFFCQIGfVTQvAAQKwqevFwx8iRFVYEDHEZfnGJwbNYI5AxxTN+UXPPY8
KjAKAQn9qxbhwI4NGVsQ721cPQkA614wjJGRwY+I9Ff0TlKTJ9rIAAbrybSSUkRrMkklhlgGM2whgEXypn16rOQXdwILCSJjeA9wpChFmQBNCn
KTIWoD+bK4yiaEAAA7'/></div>
    <h2>Server error ${ctx.request.getAttribute("javax.servlet.error.exception")!'No exception'}</h2>
    <div style="margin-top: 40px; ">
      <hr/>
      <pre>Please check server log for more detail!</pre>
    </div>
  </div>
</div>
</body>
</html>

```

This is an upgraded version of [How to setup a custom 404 handler](#) as of January 2019 because the previous one has been out dated and did not maintained for such a long time. We're trying to bring values to customers who are using and contributing to Magnolia CMS.