FatWire Professional Services

- Analysis, Design, Development, Support, QA & Recommendations

GST Site Foundation 1.2 - A Beginner's Guide

June 2011 – Version 1.3

Prepared by: Ram Sabnavis (FatWire)
ram.sabnavis@fatwire.com - +61 (0) 405 242 063

1 About This Document

This document attempts to provide a step-by-step tutorial to implement a simple, sample site based on the GST Site Foundation (GSF) Framework version 1.2.

It's intended audience are FatWire developers and the reader will greatly benefit from prior experience with the FatWire WCMS platform.

2 Table of Contents

3 About GST Site Foundation

The GST Site Foundation is a framework provided by FatWire's Global Solutions Team. It is based on the best practices and experiences of Fatwire's Professional Services with regard to implementations. This framework lays a foundation for all the necessary and usual implementation components and provides the FatWire developer, designer and architect with appropriate guidelines.

This document attempts to bring the FatWire developer, who may be a GSF novice, up to speed with GSF implementation concepts by introducing these concepts and providing a step by step tutorial for creating a GSF site.

3.1 What does GSF offer to the developer?

GSF is driven by the widespread and proven MVC architecture. The following is a list of items that GSF offers out of the box to make FatWire implementations easier and interesting:

1. Web Referenceable Asset Pattern
2. Dispatcher/Controller and Actions
3. Vanity URLs
4. Façade and Helper classes
5. JSP tags

Let us try and understand each of the above.

3.1.1 Web Referenceable Asset Pattern

Web Referenceable Assets (WRA) are assets that can be rendered and accessed through a URL.

Traditionally, Page assets were used for rendering a web page, and for navigational purposes. In addition, the system supplied Page asset type could not be modified to include any additional attributes which created problems.

GSF has answered this problem and simplified web page rendering and navigation by introducing the WRA type. Using GSF, WRA is used to render the web page, and the Page assets are now restricted to only navigational purposes.

In our tutorial below, we have created 2 different WRA types:

(a) LandingPage

(b) GSTTestArticle

The WRA can be either a Basic or a Flex asset type. However, in order to call an asset type a Web Referenceable Asset (or WRA) it should have the following attributes:

1. h1title
2. metatitle
3. metakeyword
4. metadescription
5. linktext
6. path (Standard field provided out of the box for every asset type by FatWire)
7. template (Standard field provided out of the box for every asset type by FatWire)

3.1.2 Controller and Actions

As mentioned above, GSF is based on Model-View-Controller architecture, which neatly separates presentation and business logic.

Actually, in FatWire, all requests are routed through the Content Server servlet. How, in this case, is a controller possible? GSF addresses this by including an XML CSElement called Dispatcher that can call a controller and that can assemble the model and view.

You can see an example if you jump to step 7 below where we create a CSElement of type XML which contains the following code:

```
<?xml version="1.0" ?>
<!DOCTYPE FTCS SYSTEM "futuretense_cs.dtd">

<FTCS Version="1.1">

<CALLJAVA CLASS="com.fatwire.gst.foundation.controller.action.ActionController" />

</FTCS>
```

This Dispatcher element is calling a Java class named "ActionController". This ActionController does the rest of the magic of assembling the view and model. By the way, this ActionController is provided out of the box by GSF version1.2.

What are Actions?

Actions perform logic such as Login, Search, Rendering Template etc and they can be of the following types:

1. Controller Actions
2. JSP Tag Actions

Controller Actions

These actions are called directly from the Controller Java class. The wiring of this has been done through the Spring framework. Let us take the example we have implemented in our tutorial below for search.

In Step 7, the CSElement called TopNav has the following code:

```
<form name="search" method="post" action="home?cmd=gsttest/common/Search">
       <input type="text" name="searchkeyword" value=""/>
       <input type="submit" name="search" value="Search" />

</form>
```

Look at the 'action', it has "home?**cmd**=gsttest/common/Search"

Now, the cmd instructs the controller ("ActionController" in our case) to locate the action. In gsfApplicationContext, we have configured it as follows:

```
<bean id="gsfActionLocator"
       class="com.fatwire.gst.foundation.groovy.spring.GroovyActionLocator">
       <property name="groovyLoader" ref="groovyLoader" />
              <property name="factoryClassname"
       value="com.fatwire.gst.foundation.controller.action.support.IcsBackedObjectFacto
ryTemplate" />
       </bean>
       <bean id="groovyLoader"
class="com.fatwire.gst.foundation.groovy.spring.GroovyLoader">
       </bean>
…………………………………………………………………………………….
…………………………………………………………………………………….
<bean id="gsfActionNameResolver" scope="prototype"
       class="com.fatwire.gst.foundation.controller.action.support.CommandActionNameRes
olver" />
```

The ActionController retrieves the beans with ids “gsfActionLocator”, “groovyLoader” and “gsfActionNameResolver” using Spring IOC. Then, the CommandActionNameResolver tries to match the command (cmd), specified in the URL, to a Groovy file on the file system (WEB-INF\gsf-groovy folder).

Note: Another action name resolver class, provided out of the box with GSF 1.2, is ElementActionNameResolver. This class searches for the Groovy file named after the element (Dispatcher.groovy, in our case) that calls the Controller.

Jsp Tag Actions

These actions are called by the GSF JSP tag “p:page”. Please refer to the GSF JSP Tags section below for more information.

What is Groovy?

Groovy is a dynamic language for Java similar to Python, Ruby and Perl.

“Groovy = Java - (boiler plate code + optional dynamic typing + closures + domain specific languages + builders + metaprogramming)” – an exerpt from a Groovy tutorial

Have a peek at Step 8 where we have created Groovy based actions. For example, consider common/Search.groovy:

```
1. package gsttest.common

2.  import java.text.*
3.  import COM.FutureTense.Interfaces.ICS
4.  import com.fatwire.assetapi.data.AssetId
5.  import com.fatwire.gst.foundation.controller.action.*
6.  import com.fatwire.gst.foundation.controller.annotation.*
7.  import com.fatwire.gst.foundation.facade.assetapi.asset.*
8.  import com.fatwire.gst.foundation.include.*
9.  import com.fatwire.gst.foundation.mapping.*
10. import com.fatwire.gst.foundation.wra.navigation.NavNode
11. import com.fatwire.gst.foundation.wra.navigation.NavigationHelper
12. import com.fatwire.gst.foundation.facade.search.*

13. class Search implements Action {
14     @InjectForRequest public IncludeService includeService;
15     @InjectForRequest public ScatteredAssetAccessTemplate assetDao;
16     public void handleRequest(ICS ics){
17            ics.SetVar("site", "GSTTest")
18            includeService.element("topNav", "GSTTest/TopNav").include ics
19            ics.StreamText("<h1>Search results for
20            </h1>"+ics.GetVar("searchkeyword"))
21     /* Search logic goes here */
22     }
23 }
```

Doesn’t this look similar to Java? Notice that there are no semi-colons (“;”). After the regular class import and declarations, observe lines 14 and 15 above, the GSF annotation `@InjectForRequest`.

The `@InjectForRequest` annotation will inject the objects of `IncludeService` and `ScatteredAssetAccessTemplate` making them ready for use in this action (`AnnotationInjector` class is responsible for injecting the objects specified with the annotation).

`@Mapping` is another annotation that populates the declared Java variables with the Map values in the corresponding Template or CSElement asset (MappingInjector class is responsible for injecting the map values into the variables).

3.1.3 Vanity URLs

GSF, out of the box, provides the necessary infrastructure for Vanity URLs. The components required for this feature to work are:

1. `URLAssembler` Java class - com.fatwire.gst.foundation.url.WraPathAssembler
2. `AssetEventListener` Java class - com.fatwire.gst.foundation.url.WraAssetEventListener
3. `GSTURLRegistry` database table
4. `GSTVirtualWebroot` asset (atleast one for each FatWire managed site)

Whenever a Web Referenceable Asset is created/updated/deleted, GSF registers the details appropriately into the GSTUrlRegistry table. Please ensure that the GSTVirtualWebroot asset is created before any WRAs are created. Please ensure the following steps are completed in order to have the GSTUrlRegistry table populated with the appropriate details.

1. Configure GSTVirtualWebroot prior to the creation of any WRAs
2. Provide the full Vanity URL in the "Path" field of the WRA

Note: If you create a new WRA which references a virtual webroot which does not exist, the WRA will not be indexed or appear in the GSTURLRegistry table until the virual webroot asset is created and the WRA is then edited.

3.1.4 Façade and Helper classes

GSF provides many façade and helper classes. For example, in the following line of code in GSTTest.groovy, the includeService object is internally making a call to the CallTemplate façade, which is nothing but the "render:calltemplate" tag call. GSF has provided many façade classes for the popular and commonly used FatWire tags and AssetApi. `ScatteredAssetAccessTemplate` is an example of a façade for AssetApi.

```
includeService.template (id.toString(), id, "summary")
```

In addition to the above, GSF provides helper classes to build URIs to templates/pages/blobs. The following code illustrates the usage of helper classes in the tutorial.

```
…………………………………………………….
…………………………………………………
TemplateUriBuilder pb = new TemplateUriBuilder(assetDao.currentId().getType(),
assetDao.currentId().getId().toString(), "Detail")
             Anchor anc = new Anchor()
             anc.setHref(pb.toURI(ics))
             model.add("anc", anc)
…………………………………………………….
…………………………………………………
```

In the gsttestarticle/Summary.groovy code, instead of using the render:gettempleurl tag, in the template, we can achieve the same result with the groovy action code above. All you need to do after this is put the following code in the template GSTTestArticle/Summary.

```
<a href="${anc.href}">More...</a>
```

This makes the process of referencing URIs simpler.

3.1.5 GSF JSP tags

Apart from the Java framework, the GSF also provides JSP tags. One of these tags is the “p” tag library. We are using the “p” tag library in our tutorial below.

The “p” tag library has two tags:

a. p:page
b. p:include

The “p:page” tag is used to call the Groovy actions and the “p:include” tag is used to embed calls to templates, pagelets and elements made in the corresponding Groovy code. For example, in the TestWireFrame template in our tutorial below, the usage is as follows:

```
1. <p:page action="gsttest/GSTTest">
2. ………………………………………………………
3. ………………………………………………………
4. <p:include name="topNav"/>
5. ………………………………………………………
6. ………………………………………………………

7. </p:page>
```

On line 1 above, the “p:include” tag invokes the GSTTest.groovy action located in the WEB-INF\gsf-groovy\gsttest folder. The GSTTest.groovy has the following line of code:

```
includeService.element("topNav", "GSTTest/TopNav")
```

This code has two parameters. The first parameter is the name that is referred to in the “p:include” tag. The second parameter is the element name which will be executed with render:callelement. So, at line 4 above, the element called “TopNav” will be executed and the result will be streamed.

There are other tags provided by GSF version 1.2:

`<gsf:asset-load>` – This is a facade over the asset:load tag but this tag, additionally, can specify attributes to be retrieved.

`<gsf:asset-children>` – This is a facade over the asset:children tag

`<gsf-asset-query>` – This tag querys for assets

`<gsf:tagged-list>` – Given an input gsttag attribute value, this tag returns an IList with ASSETTYPE,ASSETID as columns for any assets found that match the specified gsttag attribute value

`<gsf:tagged-assets>` – Given an input gsttag attribute value, this tag returns a Collection of AssetIds for any assets found that match the specified gsttag attribute value

`<gsf:asset-tagged-list>` – Given an input assettype and assetid, this tag returns an IList with ASSETTYPE,ASSETID as columns for any assets found that match the specified gsttag attribute value

`<gsf:navigation>` – This tag is used to retrieve the navigation nodes (page assets of subtype GSTNavLink) for a given navigation name (page asset of subtype GSTNavName)

`<gsf:root>` – This tag exposes the ics variables as JSP Expression Language (EL). Content Server variables, lists and objects can be accessed with the “cs” prefix. This tag should be placed immediately inside the <cs:ftcs> tag. Also, this tag records the dependencies for tid, seid, eid. The “p:page” tag also perform this functionality.

4 Step by Step Tutorial

4.1 The Goal

The goal of this chapter is to create and render a simple site (called GSTTest) with the following functionality:

A) Navigation to other pages
B) Non-Template pagelets/elements like "Top Nav"
C) Render Named Associations to other assets
D) Render the "Summary" pagelet of an Article with a vanity URL link to the full "Detail" page
E) Render the "Detail" page of the Article with Image
F) Search and Search Results page
G) Render a custom HTTP 404 ("Page Not Found") response

4.2 Steps to build the sample site

Please ensure that you are doing the following exercise on a GSF version 1.2 configured Jump Start Kit. To configure GSF on a clean JSK, please follow the GSF Developer Guide. This guide can be downloaded from;

http://gsf.fatwire.com/

4.2.1 Step 1: Create CS Managed Sample Site

Create a site with following details;

Name:	`GSTTest`
Description:	`GSTTest`

4.2.2 Step 2: Assign User to Site

Assign a user (say gstadmin) to the site GSTTest.

4.2.3 Step 3: Create and Share Asset Types

Enable the following asset types for the site:

1. Page
2. Template
3. CSElement
4. SiteEntry
5. AttrTypes (AttributeEditor)

Following are the asset type that can be shared from FirstSite II:

6. Media_C
7. Media_P
8. Media_A
9. Media_CD
10. Media_PD

Share the following Flex Family members from the GST site:

GST Family

Type	**Name**	**Description**	**Plural**
Flex Attribute	GSTAttribute	GST Attribute	GST Attributes
Flex Parent Defintion	GSTPDefinition	GST Parent Definition	GST Parent Definitions
Flex Definition	GSTDefintion	GST Defintion	GST Defintions
Flex Parent	GSTParent	GST Parent	GST Parents
Flex Asset	GSTVirtualWebroot	GST VirtualWebroot	GST VirtualWebroots
Flex Asset	GSTAlias	GST Alias	GST Aliases
Flex Asset	LandingPage	LandingPage	LandingPages
Flex Filter	GSTFilter	GST Filter	GST Filters

Create the following Flex Family:

GSTTest Family

Type	**Name**	**Description**	**Plural**
Flex Attribute	GSTTestAttribute	GSTTest Attribute	GSTTest Attributes
Flex Parent Defintion	GSTTestPDefinition	GSTTest Parent Definition	GSTTest PDefinitions
Flex Definition	GSTTestDefinition	GSTTest Definition	GSTTest Definitions
Flex Parent	GSTTestParent	GSTTest Parent	GSTTest Parents
Flex Asset	GSTTestArticle	GSTTest Article	GSTTest Articles
Flex Filter	GSTTestFilter	GSTTest Filter	GSTTest Filters

Note: The GSTTest Family is created in order to demonstrate that there can be as many WRA in a single site as you need. Alternatively, GSTArticle can also be created in the GST Family itself. Also, a WRA can be a basic asset type.

4.2.4 Step 4: Create Subtypes and Named Association

Ensure that the following subtypes for the Page asset type exists:

1. GSTNavLink
2. GSTNavName

Create the following named associations, if they do not exist already:

Parent Type	**Assoc Name**	**Description**	**Child Type**	**Subtypes**	**Mirror Dep**	**Single/Multi**
LandingPage	related	related	Any	LandingPage	Exists	Multi
GSTTestArticle	image	image	Media	GSTTestArticle	Exists	Multi

4.2.5 Step 5: Create Structure and Content

Share the following assets from FSII:

Asset Type	**Name**
Media_CD	FSII_Image
Media_P	FSII Article Image
Media_P	FSII Manufacturer Logos
Media_P	FSII Product Images

Create the following Page assets:

Name	Description	Subtype
MainNav	Main Nav	GSTNavName
Home	Home	GSTNavLink
Products	Products	GSTNavLink

Share the following GST Attribute assets from the GST site:

Name	Description	Value Type	Single/Multiple Value
body	Body	Text	Single
env_name	Environment Name	String	Single
env_vwebroot	Virtual Webroot	String	Single
gsttag	Tag	String	Single
h1title	h1title	String	Single
linkimage	Link Image	String	Single
linktext	Link Text	String	Single
master_vwebroot	Master Webroot	String	Single
metadescription	Metadescription	String	Single
metakeyword	Metakeyword	String	Single
metatitle	Metatitle	String	Single
popup	Popup	Int	Single
target_url	Target URL	String	Single

Share the following GST Definition assets from the GST site:

Name	Description	Attributes	Required/Optional
GSTAlias	GST Alias	h1title	Required
		metadescription	Required
		metakeyword	Required
		metatitle	Required
		target_url	Required
		Linktext	Optional
		Linkimage	Optional
		Popup	Required
GSTVirtualWebroot	GST Virtual Webroot	env_name	Required
		env_vwebroot	Optional
		master_vwebroot	Optional
LandingPage	LandingPage	metatitle	Required
		metadescription	Required
		metakeyword	Required
		h1title	Required
		linktext	Required
		body	Required

Create the following GST VirtualWebroot asset:

Name:	`GSTTestVirtualWebroot`
Description:	GSTTestVirtualWebroot
GST Definition:	GSTVirtualWebroot
Environment Name:	jsk
Virtual Webroot:	http://localhost:8280/cs
Master Webroot:	http://gsftest.fatwire.com

Note:*The Virtual Webroot is where the JSK should be listening to deliver the site.*

Create the following Landing Page assets:

1. **HomePage**

Name:	`HomePage`
Description	Home Page
Path:	http://gsftest.fatwire.com/test/home (Ensure to have Master Webroot)
GSTDefintion:	LandingPage
Metatitle:	Home
Metadescription:	Home
Metakeyword:	Home
h1title:	Home
Link Text:	Home
body:	Home Page Content

2. **Products**

Name:	`Products`
Description	Products Page
Path:	http://gsftest.fatwire.com/test/home/products (Ensure to have Master Webroot)
GSTDefintion:	LandingPage
Metatitle:	Products
Metadescription:	Products
Metakeyword:	Products
h1title:	Products
Link Text:	Products
body:	Products Page Content

Create the following GST Test Attribute assets: (Some of the attributes are similar to those of the GST Attributes as this family also has Web Referenceable Assets)

Name	**Description**	**Value Type**	**Single/Multiple Value**
abstract	abstract	Text	Single
body	Body	Text	Single
h1title	h1title	string	Single
linktext	Link Text	string	Single
metadescription	Metadescription	string	Single
metakeyword	Metakeyword	string	Single
metatitle	Metatitle	string	Single

Create the following GSTTest Definition asset:

Name	**Description**	**Attributes**	**Required/Optional**
GSTTestArticle	GSTTestArticle	h1title	Required
		metadescription	Required
		metakeyword	Required
		metatitle	Required
		linktext	Required
		abstract	Optional
		body	Optional

Create the following GSTTestArticle asset:

1. GST Test Article1

Name:	`GSTTestArticle1`
Desc:	GST Test Article1
Path:	http://gsftest.fatwire.com/test/home/products/TestArticle1
GSTTestDefintion:	GSTTestArticle
h1title:	GST Test Article1
linktext:	GST Test Article1
metadescription:	GST Test Article1
metakeyword:	GST Test Article1
metatitle:	GST Test Article1
abstract:	This is test article1 abstract
body:	This is test article 1 body

Create the following Media_C asset:

1. Article1RelatedImage

Name:	`Article1RelatedImage`
Description:	Article 1 Related Image
Media Definition:	FSII_Image
Parents:	FSII Article Images
Image File:	<<upload image>>
Alt Text:	Relate Image

4.2.6 Step 6: Create Templates

GSF moves away from the classical "FirstSite II" / "Layout" template paradigm and prefers that each template hold the the full HTML of the web page being rendered (see below the TestWireFrame template for the LandingPage and the Detail template for the GSTTestArticle examples).

Create the following templates:

4.2.6.1 TestWireFrame for LandingPage

Name:	`TestWireFrame`
Description:	WireFrame Template for LandingPage
For AssetType:	LandingPage
Usage:	Element defines a whole HTML Page
XML/JSP:	JSP
Element Logic:	

```
<%@ taglib prefix="cs" uri="futuretense_cs/ftcs1_0.tld"
%><%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"
%><%@ taglib uri="http://gst.fatwire.com/foundation/tags/gsf" prefix="gsf"
%><%@ taglib uri="http://gst.fatwire.com/foundation/tags/p" prefix="p"
%><cs:ftcs><%-- LandingPage/TestWireFrame
INPUT
OUTPUT
--%>
<p:page action="gsttest/GSTTest"><html>
       <html>
              <head>
                     <meta name="title" content='${wra.metatitle}' />
                     <meta name="description" content='${wra.metadescription}' />
                     <meta name="keyword" content='${wra.metakeyword}' />
                     <title>${wra.metatitle} | ${cs.site}</title>
              </head>
              <body>
                     <p:include name="topNav"/>
                            <p>
                     Current date is <strong><%=new java.util.Date()%></strong>
                            </p>
                            <h1>${wra.h1title}</h1>
                            <div class="articlebody">${wra.body}</div>
                            <h2>Related Articles Test</h2>
                            <c:forEach var="article" items="${related}">
                                   <p:include name="${article}" />
                            </c:forEach>
              </body>
       </html>
</p:page></cs:ftcs>
```

4.2.6.2 Summary for GSTTestArticle

Name:	Summary
Description:	Summary Template for GSTTestArticle
For AssetType:	GSTTestArticle
Usage:	Element defines a whole HTML Page
XML/JSP:	JSP
Element Logic:	

```
<%@ taglib prefix="cs" uri="futuretense_cs/ftcs1_0.tld"
%><%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"
%><%@ taglib uri="http://gst.fatwire.com/foundation/tags/gsf" prefix="gsf"
%><%@ taglib uri="http://gst.fatwire.com/foundation/tags/p" prefix="p"
%><cs:ftcs><%-- GSTTestArticle/Summary
INPUT
OUTPUT
--%>
<p:page action="gsttest/gsttestarticle/Summary">
              Article detail: ${wra.name} <br/>
             FSIIHeadline: ${wra.h1title}<br/>
             Abstract: ${wra.abstract}<br/>
             <a href="${anc.href}">More...</a>
      </p:page>
</cs:ftcs>
```

4.2.6.3 Detail for GSTTestArticle

Name:	Detail
Description:	Detail Template for GSTTestArticle
For AssetType:	GSTTestArticle
Usage:	Element defines a whole HTML Page
XML/JSP:	JSP
Element Logic:	

```
<%@ taglib prefix="cs" uri="futuretense_cs/ftcs1_0.tld"
%><%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"
%><%@ taglib uri="http://gst.fatwire.com/foundation/tags/gsf" prefix="gsf"
%><%@ taglib uri="http://gst.fatwire.com/foundation/tags/p" prefix="p"
%><cs:ftcs><%-- GSTTestArticle/Detail
INPUT
OUTPUT
--%>
<p:page action="gsttest/gsttestarticle/Detail"><html>
      <html>
             <head>
                    <meta name="title" content='${wra.metatitle}' />
                    <meta name="description" content='${wra.metadescription}' />
                    <meta name="keyword" content='${wra.metakeyword}' />
                    <title>${wra.metatitle} | ${cs.site}</title>
             </head>
             <body>
                    <p:include name="topNav"/>
                           <p>Current date is <strong><%=new
java.util.Date()%></strong>
                           </p>
                           <h1>${wra.h1title}</h1>
                           abstract:<div class="articlebody">${wra.abstract}</div>

                           <c:forEach var="relimage" items="${image}">
                                  <p:include name="${relimage}" />
                           </c:forEach>
                           <div class="articlebody">${wra.body}</div>           </body>
      </html>
</p:page>
</cs:ftcs>
```

Note: *Observe that this template assembles the TopNav again as there is no concept of a Layout template*

4.2.6.4 Detail for Media

Name:	`Detail`
Description:	Detail Template for Media
For AssetType:	Media
Usage:	Element defines a whole HTML Page
XML/JSP:	JSP
Element Logic:	

```
<%@ taglib prefix="cs" uri="futuretense_cs/ftcs1_0.tld"
%><%@ taglib prefix="ics" uri="futuretense_cs/ics.tld"
%><%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"
%><%@ taglib uri="http://gst.fatwire.com/foundation/tags/gsf" prefix="gsf"
%><%@ taglib uri="http://gst.fatwire.com/foundation/tags/p" prefix="p"
%><%@ taglib prefix="string" uri="futuretense_cs/string.tld"
%><cs:ftcs><%-- Media_C/Detail
INPUT
OUTPUT
--%>
  <p:page action="gsttest/mediac/Detail">
    <c:if test="${!empty image.src}">
      <img src="<string:stream value="${image.src}" />" class="ImageDetail"
width="${image.width}"
        height="${image.height}" alt="${image.alt}" />
    </c:if>
    <ics:clearerrno />
  </p:page>
</cs:ftcs>
```

4.2.7 Step 7: Create CSElements

Create the following CSElements assets:

4.2.7.1 GST/Dispatcher

Name:	`GST/Dispatcher`
Description:	Dispatcher of GST
XML/JSP:	**XML**
Element Logic:	

```
<?xml version="1.0" ?>
<!DOCTYPE FTCS SYSTEM "futuretense_cs.dtd">
<FTCS Version="1.1"><CALLJAVA
CLASS="com.fatwire.gst.foundation.controller.action.ActionController" /></FTCS>
```

Note:

- *Change the GST/Dispatcher to reflect the code above, if it already exists*
- *Create SiteEntry for GST/Dispatcher, if it doesn't exist already. Pass "Site=GSTTest" in resargs*

4.2.7.2 GSTTest/TopNav

Name:	`GSTTest/TopNav`
Description:	Top Navigation of GSTTest Site
XML/JSP:	JSP
Element Logic:	

```
<%@ taglib prefix="cs" uri="futuretense_cs/ftcs1_0.tld"
%><%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"
%><%@ taglib uri="http://gst.fatwire.com/foundation/tags/gsf" prefix="gsf"
%><%@ taglib uri="http://gst.fatwire.com/foundation/tags/p" prefix="p"
%><cs:ftcs><%-- TopNav
INPUT
OUTPUT
--%>
<p:page action="gsttest/common/TopNav">
		<div id="nav">
			<ul>
				<c:forEach var="kid" items="${MainNav}">
					<li><a href='${kid.url}'>${kid.linktext}</a></li>
					<ul>
					<c:forEach var="kid2" items="${kid.children}">
						<li><a
href='${kid2.url}'>${kid2.linktext}</a></li>
					</c:forEach>
					</ul>
				</c:forEach>
			</ul>
		</div>
		<form name="search" method="post"
action="home?cmd=gsttest/common/Search">
		<input type="text" name="searchkeyword" value=""/>
		<input type="submit" name="search" value="Search" />
		</form>
	</p:page>
</cs:ftcs>
```

4.2.7.3 GSTTest/ErrorHandler/404.jsp

Name:	`GSTTest/ErrorHandler/404`
Description:	Custom HTTP 404 Error Response for the GSTTest Site
XML/JSP:	JSP
Element Logic:	

```
<%@ taglib prefix="cs" uri="futuretense_cs/ftcs1_0.tld"
%><cs:ftcs><%--GSTTest/ErrorHandler/404
INPUT
OUTPUT
--%>
<h1> This is custom 404 Page </h1>
</cs:ftcs>
```

4.2.7.4 GST/ErrorHandler/404.jsp

Name:	`GST/ErrorHandler/404`
Description:	General HTTP 404 Error Response
XML/JSP:	JSP
Element Logic:	

```
<%@ taglib prefix="cs" uri="futuretense_cs/ftcs1_0.tld"
%><cs:ftcs><%-- GST/ErrorHandler/404
INPUT
OUTPUT
--%>
<h1>This is GST 404</h1>
</cs:ftcs>
```

4.2.8 Step 8: Create Groovy Actions Files

Create "gsttest" folder under "<<webapps>>../WEB-INF/gsf-groovy/" and create the following Groovy files:

4.2.8.1 GSTTest.groovy

```
package gsttest

import java.text.*
import COM.FutureTense.Interfaces.ICS
import com.fatwire.assetapi.data.AssetId
import com.fatwire.gst.foundation.controller.action.*
import com.fatwire.gst.foundation.controller.annotation.*
import com.fatwire.gst.foundation.facade.assetapi.asset.*
import com.fatwire.gst.foundation.include.*
import com.fatwire.gst.foundation.mapping.*
import com.fatwire.gst.foundation.wra.navigation.NavNode
import com.fatwire.gst.foundation.wra.navigation.NavigationHelper

class GSTTest implements Action {
        @InjectForRequest public IncludeService includeService;
        @InjectForRequest public ScatteredAssetAccessTemplate assetDao;
        @InjectForRequest public Model model;
        @InjectForRequest public NavigationHelper navHelper;
        public void handleRequest(ICS ics){


        model.add("wra",assetDao.readCurrent("metatitle","metadescription","metakeyword"
,"h1title","linktext","body"));
                includeService.element("topNav", "GSTTest/TopNav")
                Collection<AssetId> related=assetDao.readAssociatedAssetIds ("related");
                for(AssetId id:related){
                        model.list ("related", id.toString());
        includeService.template (id.toString(), id, "Summary")
                }
        }
}
```

4.2.8.2 common/TopNav.groovy

```
package gsttest.common

import java.text.*
import COM.FutureTense.Interfaces.ICS
import com.fatwire.assetapi.data.AssetId
import com.fatwire.gst.foundation.controller.action.*
import com.fatwire.gst.foundation.controller.annotation.*
import com.fatwire.gst.foundation.facade.assetapi.asset.*
import com.fatwire.gst.foundation.include.*
import com.fatwire.gst.foundation.mapping.*
import com.fatwire.gst.foundation.wra.navigation.NavNode
import com.fatwire.gst.foundation.wra.navigation.NavigationHelper

class TopNav implements Action {
        @InjectForRequest public Model model;
        @InjectForRequest public NavigationHelper navHelper;
        public void handleRequest(ICS ics){
                NavNode node = navHelper.getSitePlanByPage(-1,"MainNav");
                if(node !=null) model.add("MainNav",node.getChildren());
        }
}
```

4.2.8.3 common/Search.groovy

```
package gsttest.common

import java.text.*
import COM.FutureTense.Interfaces.ICS
import com.fatwire.assetapi.data.AssetId
import com.fatwire.gst.foundation.controller.action.*
import com.fatwire.gst.foundation.controller.annotation.*
import com.fatwire.gst.foundation.facade.assetapi.asset.*
import com.fatwire.gst.foundation.include.*
import com.fatwire.gst.foundation.mapping.*
import com.fatwire.gst.foundation.wra.navigation.NavNode
import com.fatwire.gst.foundation.wra.navigation.NavigationHelper
import com.fatwire.gst.foundation.facade.search.*
class Search implements Action {
       @InjectForRequest public IncludeService includeService;
       @InjectForRequest public ScatteredAssetAccessTemplate assetDao;
       public void handleRequest(ICS ics){
              ics.SetVar("site", "GSTTest")
              includeService.element("topNav", "GSTTest/TopNav").include ics
              ics.StreamText("<h1>Search results for
</h1>"+ics.GetVar("searchkeyword"))
       /* Search logic goes here */
       }
}
```

4.2.8.4 gsttestarticle/Summary.groovy

```
package gsttest.gsttestarticle

import java.text.*
import COM.FutureTense.Interfaces.ICS
import com.fatwire.assetapi.data.AssetId
import com.fatwire.gst.foundation.controller.action.*
import com.fatwire.gst.foundation.controller.annotation.*
import com.fatwire.gst.foundation.facade.assetapi.asset.*
import com.fatwire.gst.foundation.include.*
import com.fatwire.gst.foundation.mapping.*
import com.fatwire.gst.foundation.wra.navigation.NavNode
import com.fatwire.gst.foundation.wra.navigation.NavigationHelper
import com.fatwire.gst.foundation.html.Anchor
import com.fatwire.gst.foundation.facade.uri.TemplateUriBuilder
class Summary implements Action {
       @InjectForRequest public IncludeService includeService;
       @InjectForRequest public ScatteredAssetAccessTemplate assetDao;
       @InjectForRequest public Model model;
       @InjectForRequest public NavigationHelper navHelper;

       public void handleRequest(ICS ics){
       model.add("wra",assetDao.readCurrent("metatitle","metadescription","metakeyword"
,"h1title","linktext","body","abstract"));
              TemplateUriBuilder pb = new
TemplateUriBuilder(assetDao.currentId().getType(),
assetDao.currentId().getId().toString(), "Detail")
              Anchor anc = new Anchor()
              anc.setHref(pb.toURI(ics))
              model.add("anc", anc)
       }
}
```

4.2.8.5 gsttestarticle/Detail.groovy

```
package gsttest.gsttestarticle

import java.text.*
import COM.FutureTense.Interfaces.ICS
import com.fatwire.assetapi.data.AssetId
import com.fatwire.gst.foundation.controller.action.*
import com.fatwire.gst.foundation.controller.annotation.*
import com.fatwire.gst.foundation.facade.assetapi.asset.*
import com.fatwire.gst.foundation.include.*
import com.fatwire.gst.foundation.mapping.*
import com.fatwire.gst.foundation.wra.navigation.NavNode
import com.fatwire.gst.foundation.wra.navigation.NavigationHelper
class Detail implements Action {
       @InjectForRequest public IncludeService includeService;
       @InjectForRequest public ScatteredAssetAccessTemplate assetDao;
       @InjectForRequest public Model model;
       @InjectForRequest public NavigationHelper navHelper;
       public void handleRequest(ICS ics){
       model.add("wra",assetDao.readCurrent("metatitle","metadescription","metakeyword"
,"h1title","linktext","body","abstract"));

              //Call TopNav CSElement
              includeService.element("topNav", "GSTTest/TopNav")
              //Call Detail Template for Media
              Collection<AssetId> image=assetDao.readAssociatedAssetIds ("image");
              for(AssetId id:image){
                     model.list ("image", id.toString());
                     def summary = includeService.template (id.toString(), id, "Detail")
              }
       }
}
```

4.2.8.6 mediac/Detail.groovy

```
package gsttest.mediac

import COM.FutureTense.Interfaces.ICS;
import org.apache.commons.lang.StringUtils
import COM.FutureTense.Interfaces.ICS
import com.fatwire.gst.foundation.controller.action.Action
import com.fatwire.gst.foundation.controller.action.Model
import com.fatwire.gst.foundation.controller.annotation.InjectForRequest
import com.fatwire.gst.foundation.controller.annotation.Mapping
import com.fatwire.gst.foundation.controller.annotation.Mapping.Match
import com.fatwire.gst.foundation.facade.assetapi.asset.ScatteredAssetAccessTemplate
import com.fatwire.gst.foundation.facade.assetapi.asset.TemplateAsset
import com.fatwire.gst.foundation.facade.assetapi.asset.TemplateAssetMapper
import com.fatwire.gst.foundation.facade.uri.BlobUriBuilder
import com.fatwire.gst.foundation.html.Img
import com.fatwire.gst.foundation.include.IncludeService

class Detail implements Action {
    @InjectForRequest public ScatteredAssetAccessTemplate assetDao;
    @InjectForRequest public Model model;
    /* The following code works if the calling Template Asset's MAP has been populated
as follows
    @Mapping(value="ImageFileAttrName", match=Match.right) public String
ImageFileAttrName
    @Mapping(value="ImageMimeTypeAttrName", match=Match.right) public String
ImageMimeTypeAttrName
    @Mapping(value="ImageWidthAttrName", match=Match.right) public String
ImageWidthAttrName
```

```
    @Mapping(value="ImageHeightAttrName", match=Match.right) public String
ImageHeightAttrName
    @Mapping(value="AltTextAttrName", match=Match.right) public String AltTextAttrName
    */
       public String AltTextAttrName = "FSII_AltText"
       public String ImageFileAttrName = "FSII_ImageFile"
       public String ImageMimeTypeAttrName = "FSII_ImageMimeType"
       public String ImageWidthAttrName = "FSII_ImageWidth"
       public String ImageHeightAttrName = "FSII_ImageHeight"
    @Override
    public void handleRequest(ICS ics) {
        TemplateAssetMapper mapper = new TemplateAssetMapper();
        TemplateAsset asset = assetDao.readAsset(assetDao.currentId(),
mapper,ImageMimeTypeAttrName,ImageWidthAttrName,ImageHeightAttrName,AltTextAttrName,Ima
geFileAttrName);
        BlobUriBuilder ub = new BlobUriBuilder(asset.asBlob(ImageFileAttrName));
        ub.mimeType(asset.asString(ImageMimeTypeAttrName))
        Img img = new Img();
        img.setSrc(ub.toURI(ics));
        img.setWidth asset.asString(ImageWidthAttrName)
        img.setHeight asset.asString(ImageHeightAttrName)
        String alt = asset.asString(AltTextAttrName);
        if(StringUtils.isBlank(alt)){
            alt="Content Server Image"
        }
        img.setAlt alt
        model.add("image",img);
    }
}
```

Note:

- *Ensure that, in futuretense.ini, cs.sitepreview property is set to either “contentmanagement” (on authoring environment) or blank (on Delivery environment). If this property is set to “disabled” then navHelper.getSitePlanByPages (referred in common/TopNav.groovy above) may not work.*

4.2.9 Step 9: Place Pages into Site Plan

Place the Page assets into the Site Plan as follows:

```
MainNav (subtype = GSTNavName)
         |
         |-----Home (subtype = GSTNavLink)
                  |
                  |----- Products (subtype = GSTNavLink)
```

4.2.10 Step 10: Wiring the assets

Populate associations as follows:

Parent (AssetType:AssetName)	Child (AssetType: AssetName)	Named/Unnamed
Page:Home	LandingPage:Home	Unnamed
Page:Products	LandingPage:Products	Unnamed
LandingPage:Products	GSTTestArticle:GSTTestArticle1	related
GSTTestArticle:GSTTestArticle1	Media_C:Article1RelatedImage	Image

Note: Ensure that the *LandingPage*, *GSTTestArticle* and *Media_C* have been enabled to be a child types.

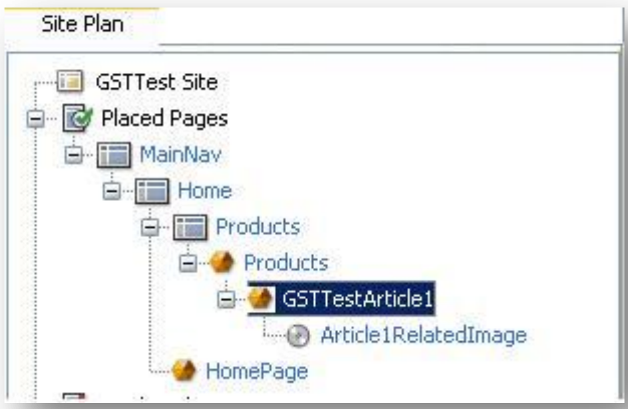
Edit the following assets to assign the templates:

Asset Type	Asset Name	Template
LandingPage	Home	TestWireFrame
LandingPage	Products	TestWireFrame
GSTTestArticle	GSTTestArticle1	Detail

Note:

- *A HTTP 500 error will be thrown if the templates are not associated*
- *Enabling the following loggers helps to debug the request*
  - `com.fatwire.gst.foundation.url.WraPathAssembler=TRACE`
  - `com.fatwire.gst.foundation.controller=TRACE`

The following screenshot illustrates the placement of the Page assets and their asset associations



4.2.11 Step 11: Configure Tuckey URLRewrite

Configure Tuckey in <<webapps/cs/>>`\WEB-INF\urlrewrite.xml` as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE urlrewrite PUBLIC "-//tuckey.org//DTD UrlRewrite 3.2//EN"
        "http://www.tuckey.org/res/dtds/urlrewrite3.2.dtd">
<urlrewrite use-query-string="true">
       <rule match-type="regex">
             <condition type="port">8280</condition>
             <from>/Satellite</from>
             <to type="forward" last="true">-</to>
       </rule>
       <rule match-type="regex">
             <condition type="port">8280</condition>
             <from>/CookieServer</from>
             <to type="forward" last="true">-</to>
       </rule>
       <rule match-type="regex">
             <condition type="port">8280</condition>

       <from>/(FCKeditor)|(NetImaging)|(flash)|(js)|(resources)|(wemresources)|(ImageEd
itor)|(custom)|(schema)|(Xcelerate)|(html)|(sites)|(Master)|(images)|(login)|(skins)|(M
ediaPlayer)|(cachetool)|(export)|(remoteimages)|(userfiles)/.*</from>
             <to type="forward" last="true">-</to>
       </rule>
       <rule match-type="regex">
             <condition type="port">8280</condition>
             <condition type="method">GET</condition>
             <from>/(.*)</from>
             <to type="forward">/Satellite?virtual-
webroot=http://localhost:8280/cs&amp;pagename=GST/Dispatcher&amp;url-
path=/$1&amp;%{query-string}</to>
       </rule>
       <rule match-type="regex">
             <condition type="port">8280</condition>
             <condition type="method">POST</condition>
             <from>/(.*)</from>
             <to type="forward">/Satellite?virtual-
webroot=http://localhost:8280/cs&amp;pagename=GST/Dispatcher&amp;url-path=/$1</to>
       </rule>
</urlrewrite>
```

Note:

- *It is expected that the Tuckey URLRewrite filter is configured in the* `web.xml` *of the CS web application in the Jump Start Kit*
- *Tuckey can be replaced with Apache's mod_rewrite module*

4.2.12 Step 12: Configure GSF Application Context

Configure `gsfApplicationContext.xml` as follows

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN"
"http://www.springframework.org/dtd/spring-beans-2.0.dtd">
<beans>
       <!-- INSTALLATION: Be sure to add this to your WEB-INF/web.xml file: <context-
param>
              <param-name>contextConfigLocation</param-name> <param-value>/WEB-
INF/applicationContext.xml,/WEB-INF/gsfApplicationContext.xml</param-value>
              </context-param> USAGE: Be sure to set any actions that need to be
sateful
              to have a scope="prototype" in order that they are created as new
instances. -->

<bean id="gsfActionLocator"
       class="com.fatwire.gst.foundation.groovy.spring.GroovyActionLocator">
       <property name="groovyLoader" ref="groovyLoader" />
              <property name="factoryClassname"
       value="com.fatwire.gst.foundation.controller.action.support.IcsBackedObjectFacto
ryTemplate" />
       </bean>

       <bean id="groovyLoader"
class="com.fatwire.gst.foundation.groovy.spring.GroovyLoader">
       </bean>

       <bean id="gsfRenderPage" scope="prototype"
              class="com.fatwire.gst.foundation.controller.action.RenderPage" />

       <bean id="gsfActionNameResolver" scope="prototype"

       class="com.fatwire.gst.foundation.controller.action.support.CommandActionNameRes
olver" />
</beans>
```

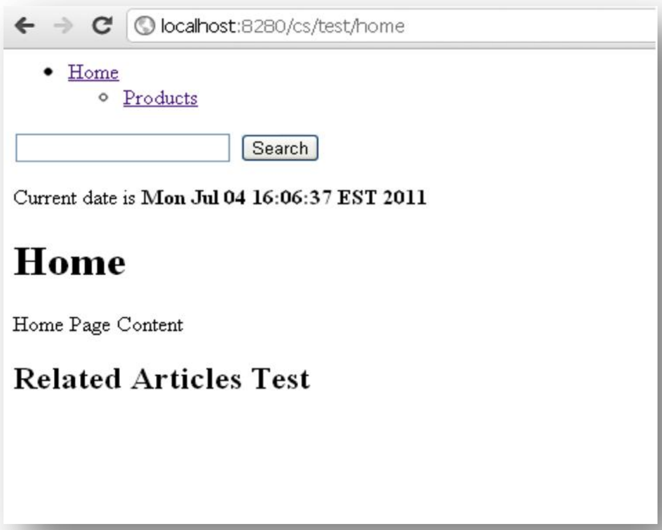
Note:

- *`gsfApplicationContext.xml` should already be configured in `WEB-INF` folder. Please check if it resembles the code above.*
- *`gsfActionLocator` object is injected and is referred by `ActionController`*
- *`gsfRenderPage` is responsible for calling/rendering the template associated to the WRA*
- *`gsfActionNameResolver` is responsible for identifying the actions specified ie., `CommandActionNameResolver` executes groovy file by the name specified as “cmd” (querystring) variable and `ElementNameActionNameResolver` executes the groovy file with the name of the CSElement that has invoked the controller*

4.3 GSTTest sample site rendered

The following should be the results of your code so far:

4.3.1 Home Page with navigation



localhost:8280/cs/test/home

- Home
  - Products

Search

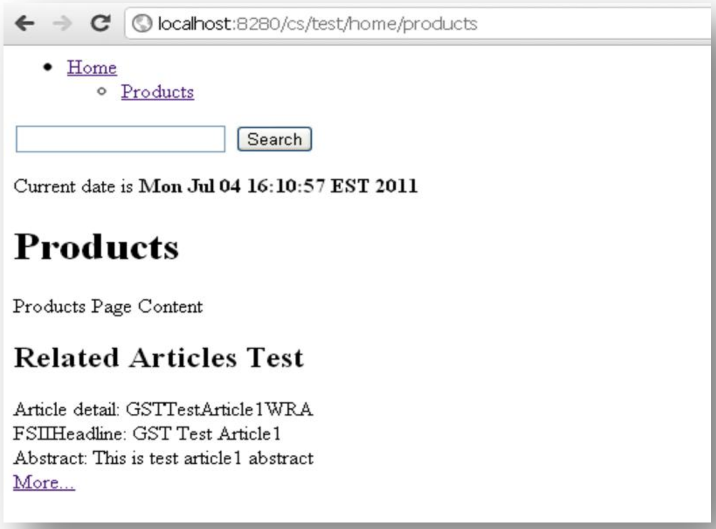
Current date is **Mon Jul 04 16:06:37 EST 2011**

Home

Home Page Content

Related Articles Test

4.3.2 Products Page with Article Summary



localhost:8280/cs/test/home/products

- Home
  - Products

Search

Current date is **Mon Jul 04 16:10:57 EST 2011**

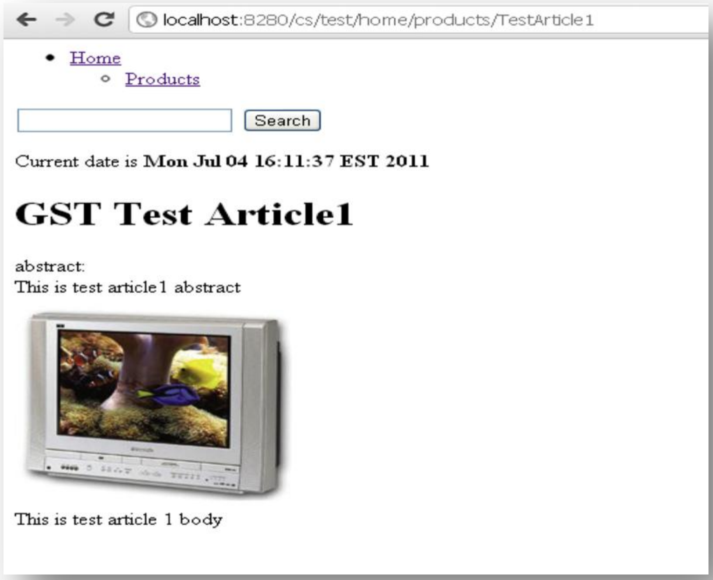
Products

Products Page Content

Related Articles Test

Article detail: GSTTestArticle1WRA
FSIIHeadline: GST Test Article1
Abstract: This is test article1 abstract
More...

4.3.3 Article Detail with associated Image



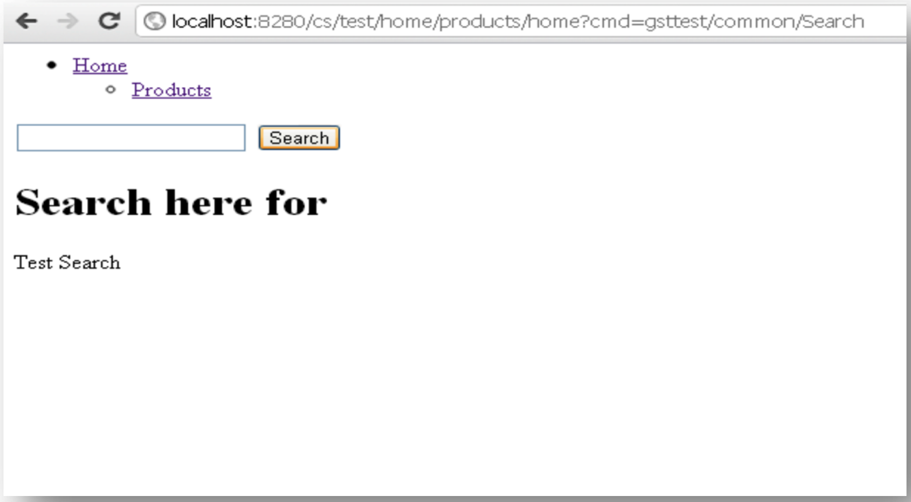
Current date is **Mon Jul 04 16:11:37 EST 2011**

GST Test Article1

abstract:
This is test article1 abstract

This is test article 1 body

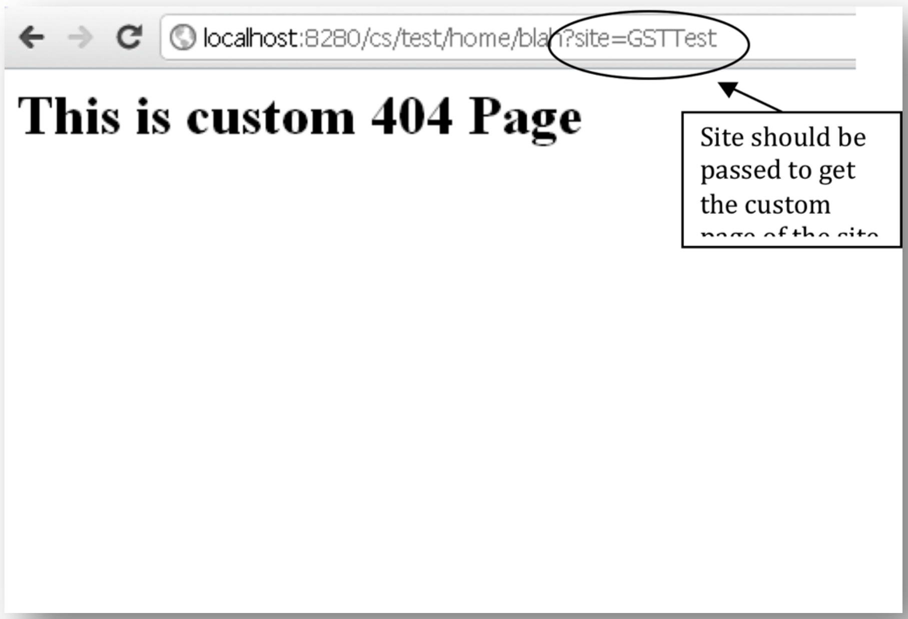
4.3.4 Search Results Page


localhost:8280/cs/test/home/products/home?cmd=gsttest/common/Search
Home
Products
Search

Search here for

Test Search

4.3.5 Custom HTTP 404 resonse

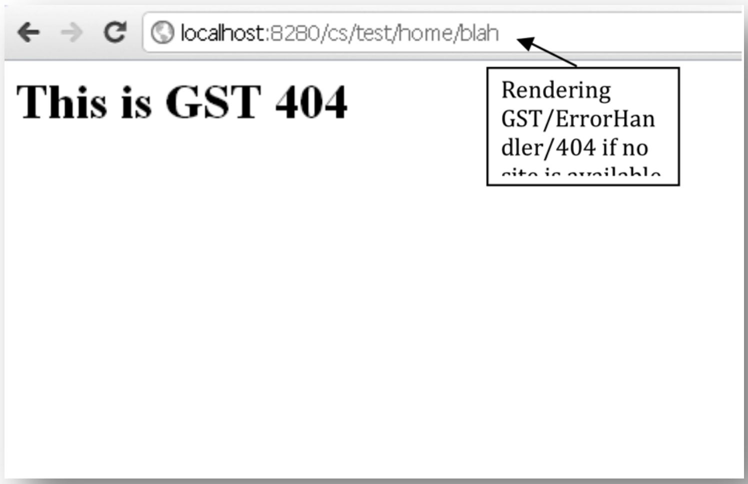


localhost:8280/cs/test/home/blah?site=GSTTest

This is custom 404 Page

Site should be passed to get the custom page of the site

4.3.6 General GST HTTP 404 response (if site value is not available)



localhost:8280/cs/test/home/blah

This is GST 404

Rendering GST/ErrorHandler/404 if no site is available