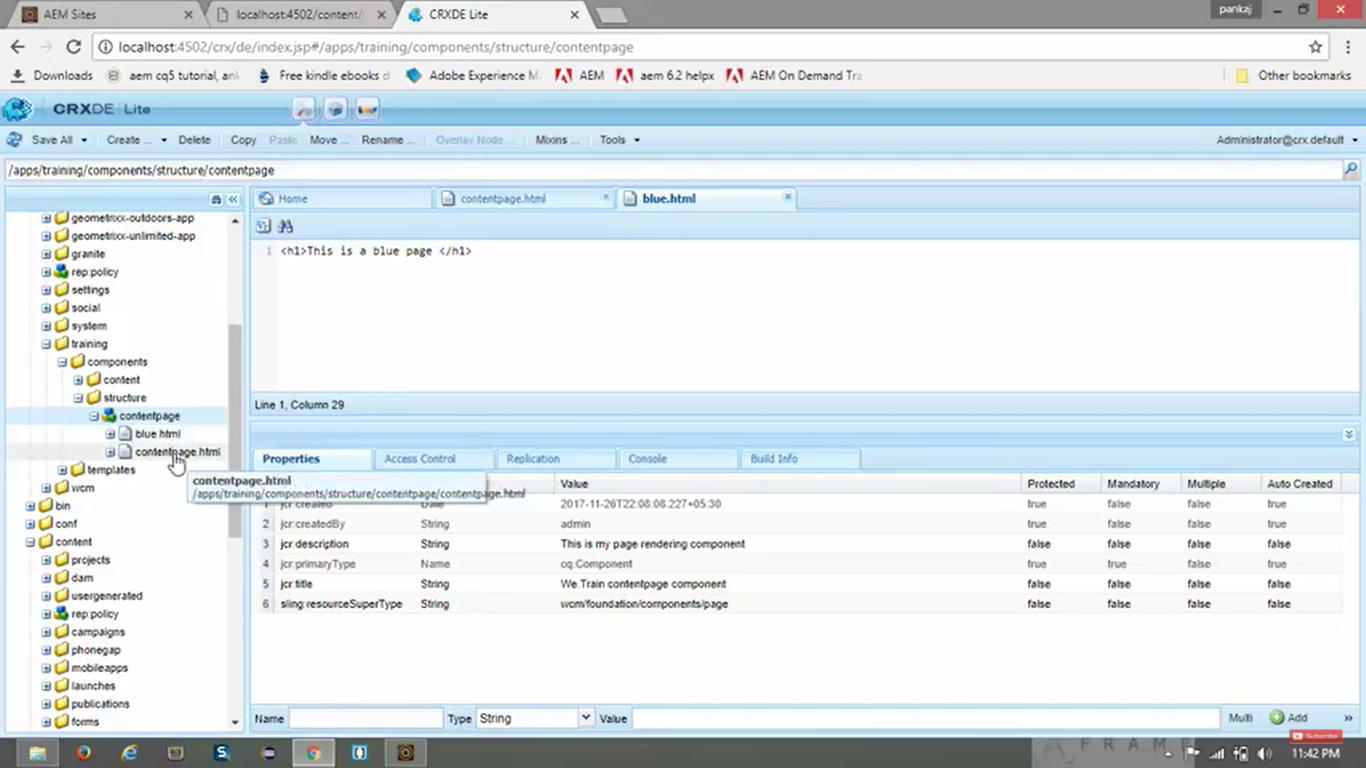
# 16. Apache sling selector manipulation:

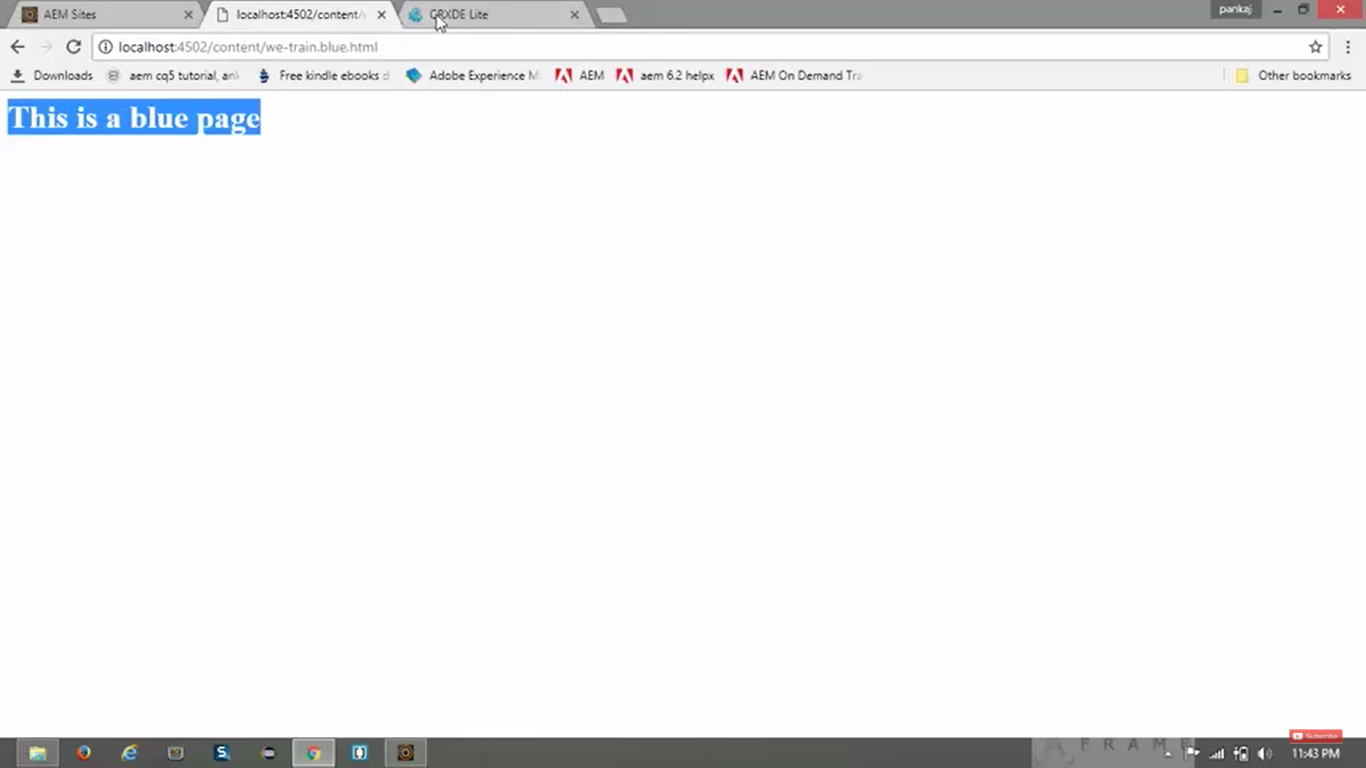
1. Go to **AEM instance** --> **Sites** --> Click on **We.Train** website which we have recently created.
2. Click on **Edit** icon, so that the *Page will open* in **Edit mode**.
3. This will open <http://localhost:4502/editor.html/content/we-train.html> because we are logged in to **AUTHORING environment**.
4. Remove **editor.html** from URL to make <http://localhost:4502/content/we-train.html>.
5. Let’s consider 1 more example given below
6. Go to **CRX/DE**. Type **/apps/training/components/structure/contentpage** on address bar.
7. Now, **Right Click** on **contentpage**, select **Create**, select sub-option **Create File**.
8. Enter file name **blue.html**. Click on **OK** button. Then click on **Save** button on top of the page.
9. Enter file content as “<h1> This is blue page </h1>”. Then click on **Save** button on top of the page.
10. Now if we open URL <http://localhost:4502/content/we-train.html>, it will search for node “content/we-train”.
11. Next, it will check **sling:resourceType** property of **jcr:content** node of this node **/content/we-train**.

|  |  |
| --- | --- |
| **Field** | **Value** |
| sling:resourceType | training/components/structure/contentpage |

1. Now, there are 2 components, **blue.html** and **contentpage.html**.
2. So, since we put **URL** <http://localhost:4502/content/we-train.html> in browser it will open default page with name same as node name. That means, it will open **contentpage.html** under node **training/components/structure/contentpage**.
3. But if we want to open blue.html page under that same node. Then we have to provide selector in the URL. That is, <http://localhost:4502/content/we-train.blue.html>
4. This will node open default page **contentpage.html**, but it will open **blue.html** under same parent node, because we have provided the selector as **blue.html** at the end of the URL this time.
5. **Conclusion**: **Selector** has *higher priority* than default page.







# 17. Restrict the use of template using allowedPaths and allowedTemplate properties:

1. **allowedPaths** property:

* It defines where the template can be used to create pages.
* This is applied as a property value on Template page.

**Example:**

|  |  |
| --- | --- |
| **Value** | **Meaning** |
| /content(/.\*)? | Template will be available for creating page in **any** website |
| /content(we-train/.\*)? | Template will be available for creating page in **only we-train** website |

1. Go to **CRX/DE**. Put URL in address bar as **/apps/training/templates/contentpage**
2. See **allowedPaths** property of this node “contentpage”/ with value “/content(/.\*)?”
3. Now go to **AEM --> Sites** --> Click on **We-Train** website --> Click on **Create** button on top -->

Click on **Page** option.

Here you can see “We-Train Content Page” **Template** *can be used* to create new page.

1. Create a **Page** named **childpage** within **/apps/training/templates**.
2. In **Property Pane**, set below **Property**, for same **childpage**:

|  |  |
| --- | --- |
| **Value** | **Meaning** |
| /content(we-train/.\*)? | Template will be available for creating page in only we-train website |

1. Now go **AEM --> Sites** --> Click on **Create** button on top --> Click on **Page** option.
2. Here you cannot see that **childpage** template.
3. Now go **AEM --> Sites** --> Click on **We-Train website** --> Click on **Create** button on top -->

Click on **Page** option.

1. Here you can see the **childpage** template is available to use
2. **allowedTemplate** property:

* It specifies which templates can be used within a specified path.
* This is applied as a property value on Template page.

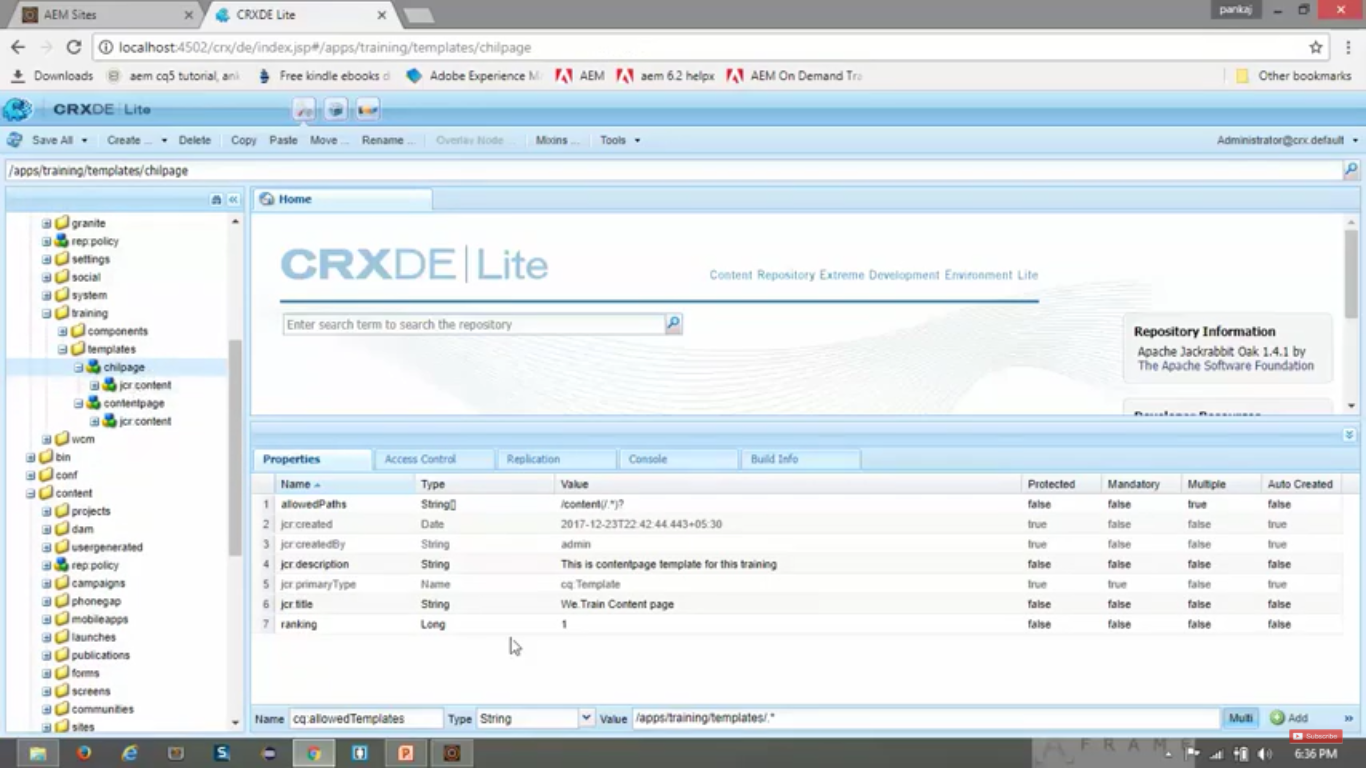
1. If you restrict specific set of templates to be used for a specific website, then you can use **allowedTemplate** property.
2. Go to **CRX/DE**. Type **/content/we-train/jcr:content** in address bar. (**jcr:content** of website).
3. **Create** *new* **Property** with following details, at the bottom most section in **Properties Pane**.

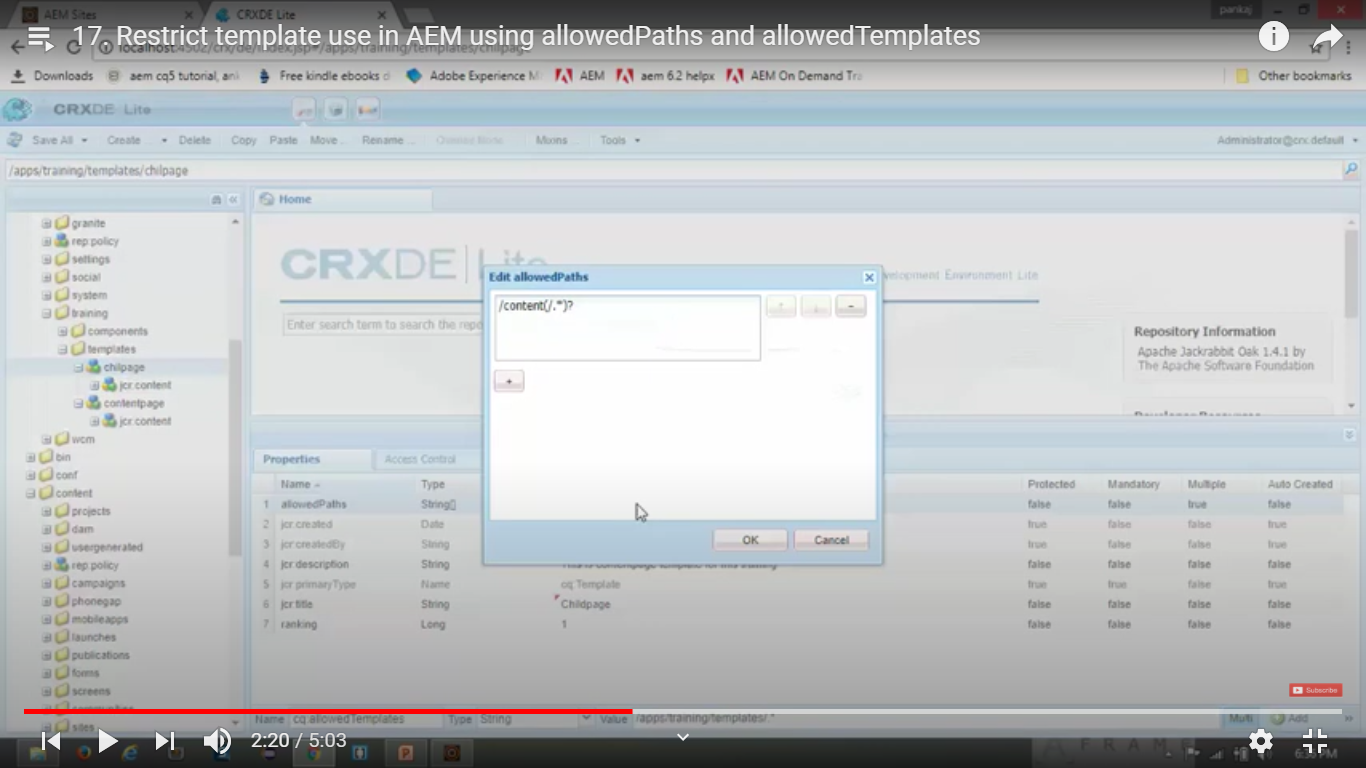
|  |  |
| --- | --- |
| **Field Name** | **Field Value** |
| Name | cq:allowedTemplates |
| Type | String |
| Click on --> | “Multi” button |
| Value | /apps/training/templates/\* |

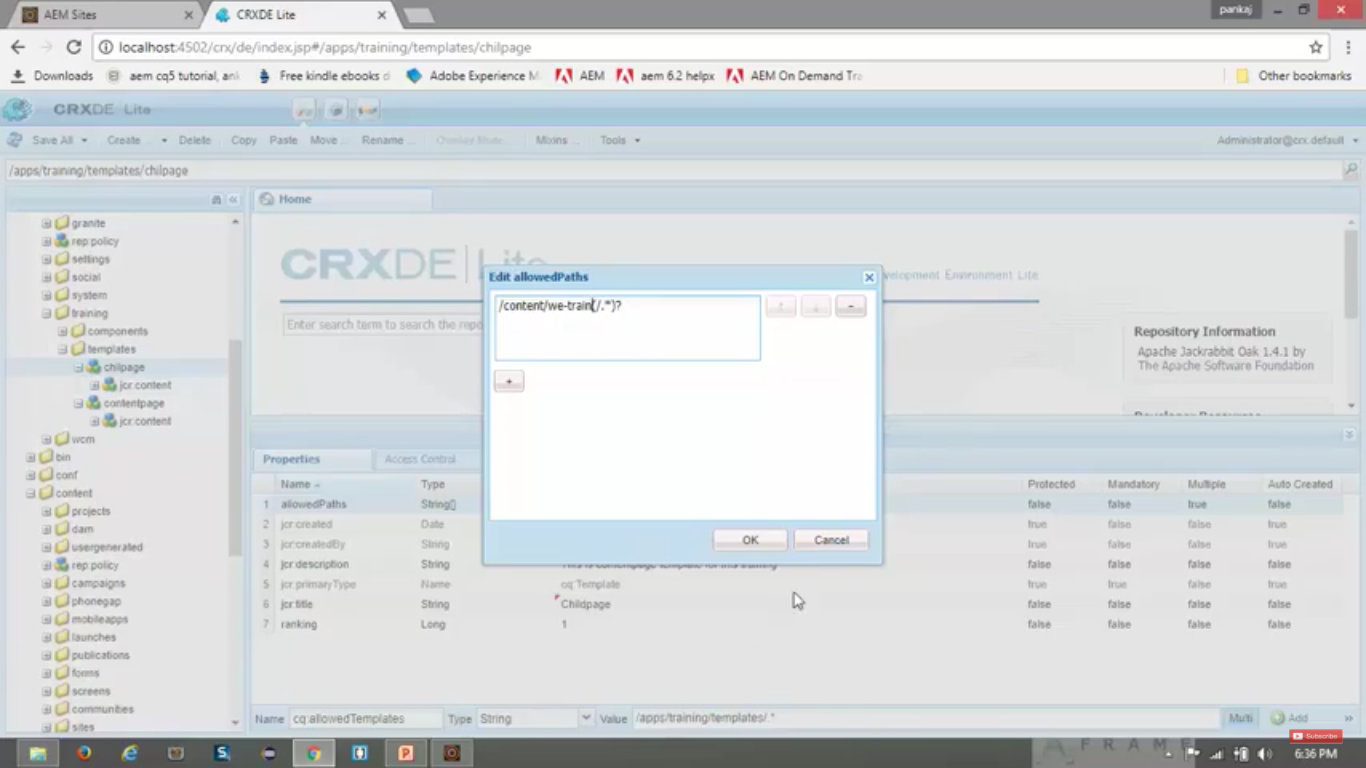
1. Click on **Add** button -> Click on **OK** button on dialog box --> Click on **Save All** button on Toolbar.
2. ***To test***, go to **AEM** --> **Sites** --> Click on **We-Train website** --> Click on **Create** button -->

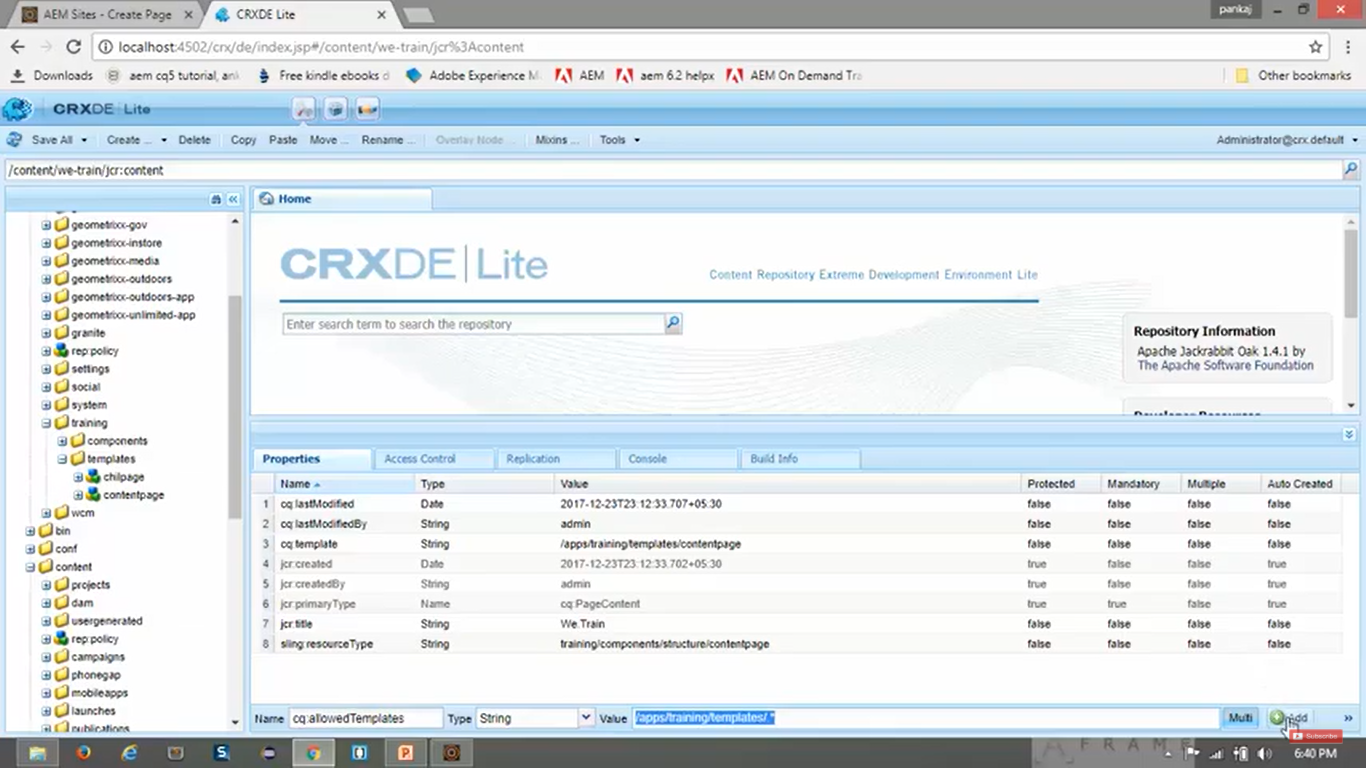
Click on **Page**.

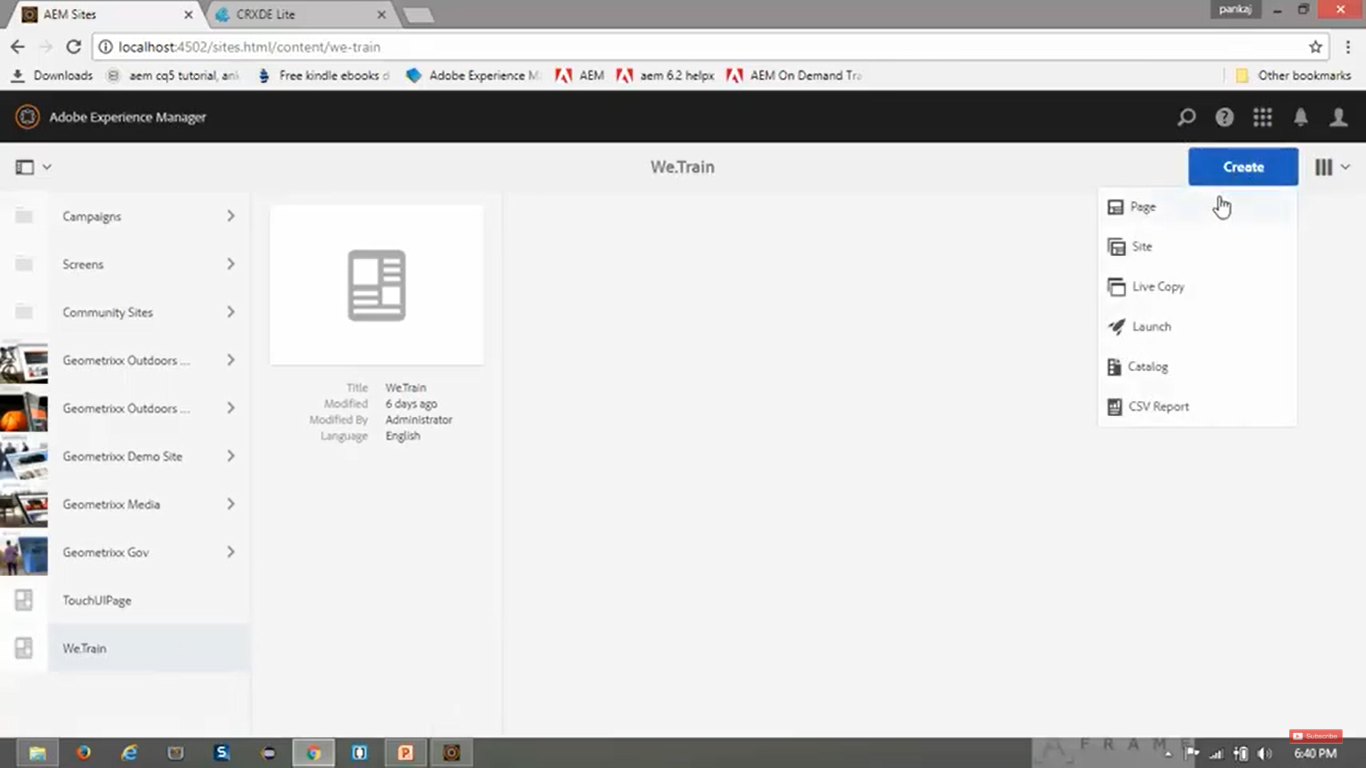
1. Now you will see only 2 templates which are mentioned in the specified address created under **/apps/training/templates/**

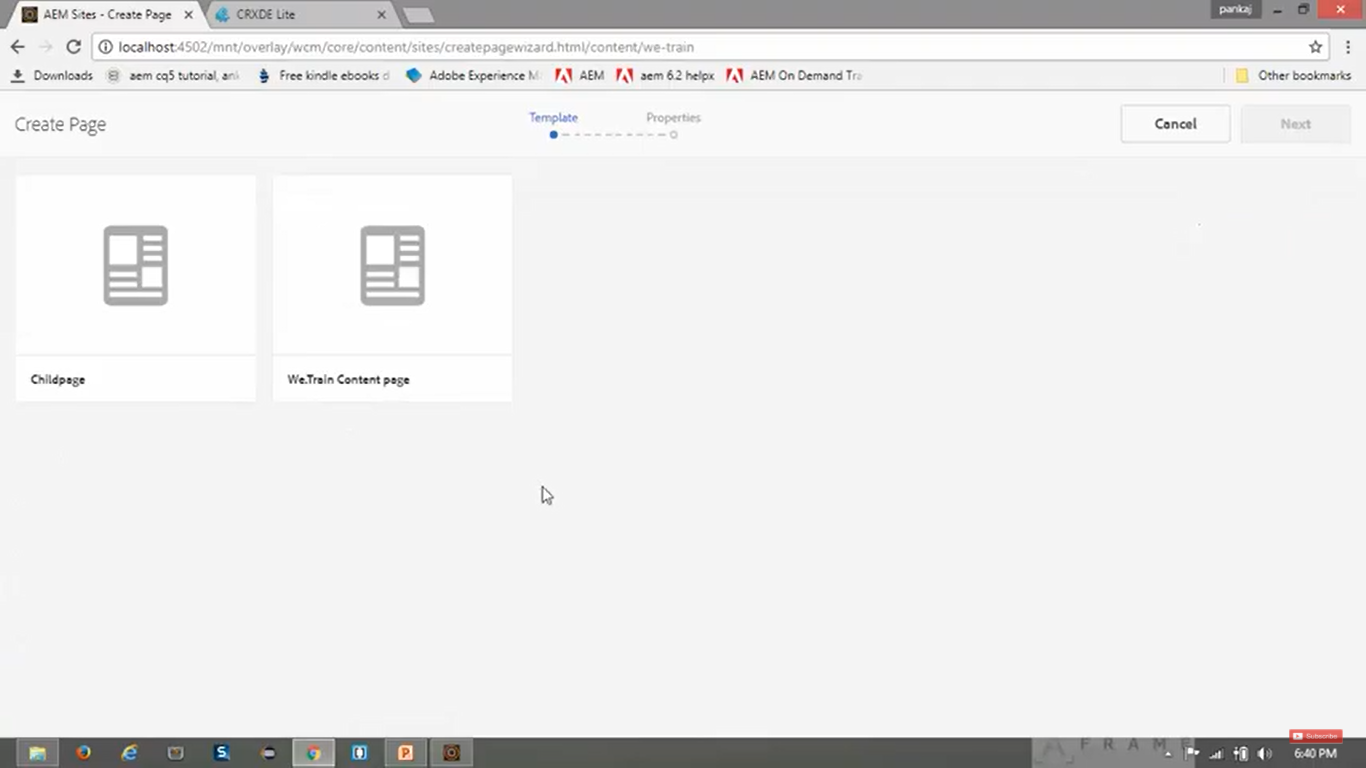










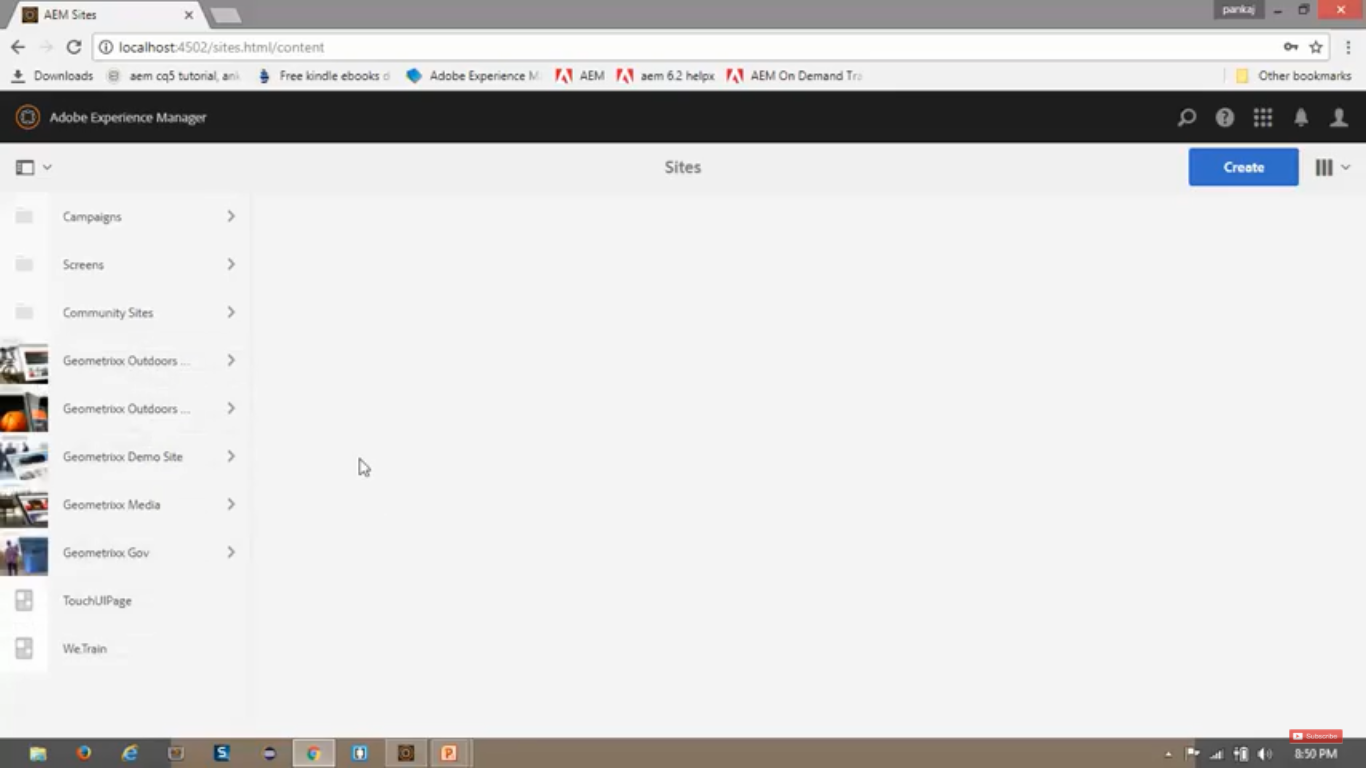


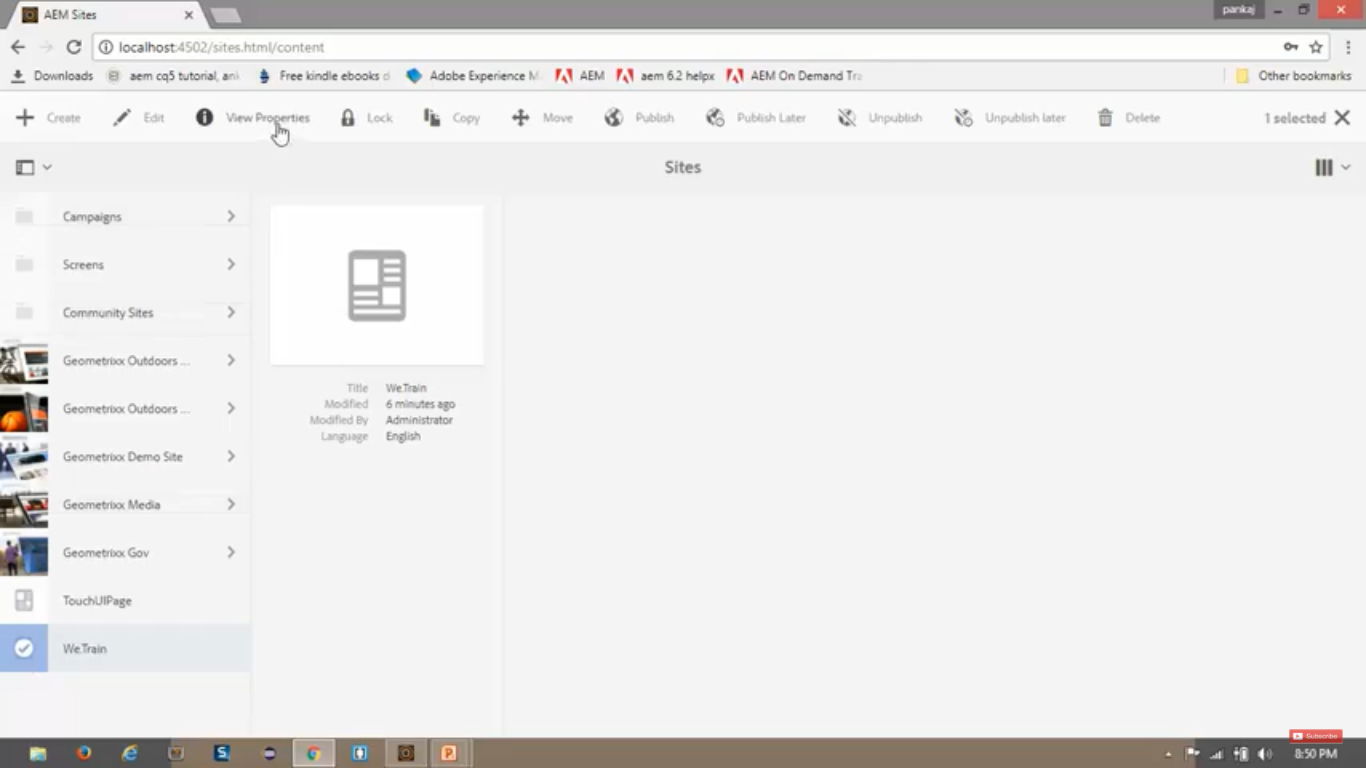
# 18. Adding Thumbnail to page in AEM:

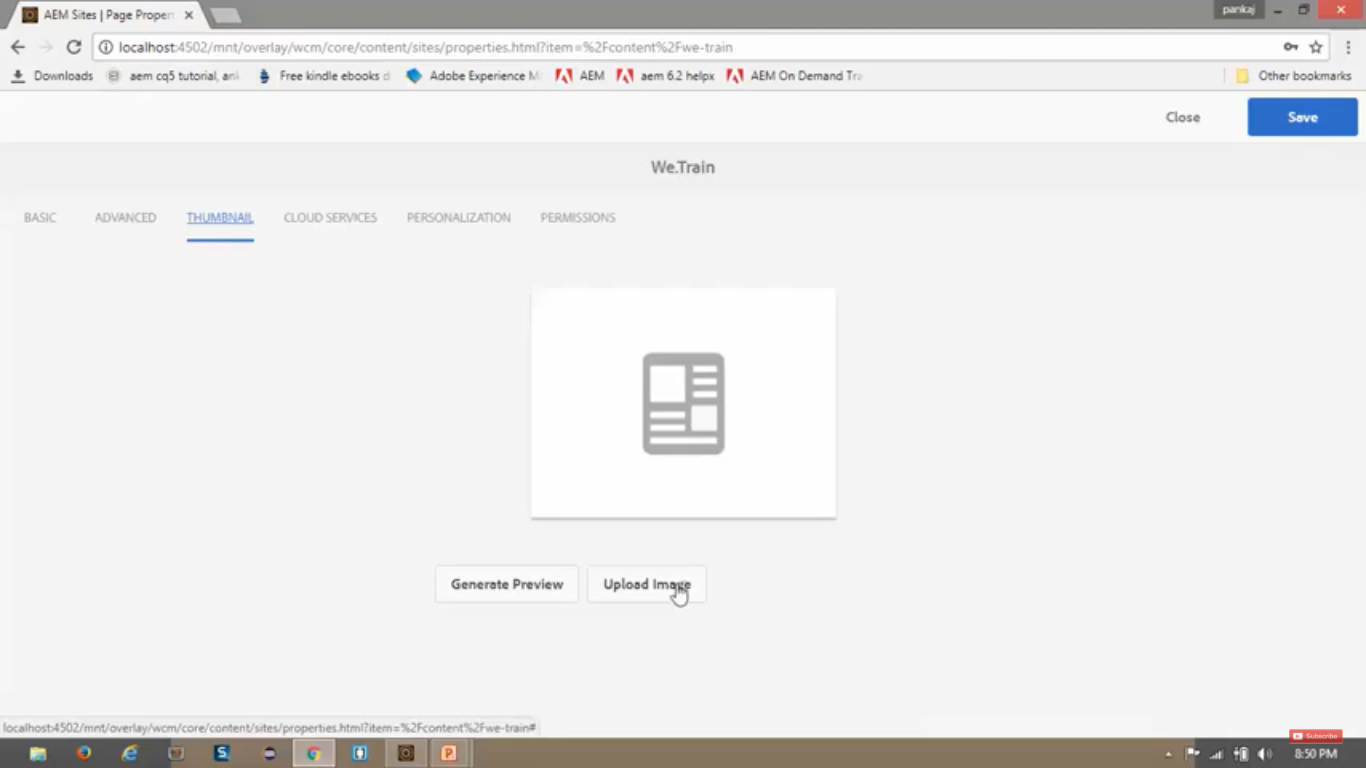
1. Go to **AEM --> Sites**. Here You will see your newly created website “We.Train”
2. You can see there is small thumbnail image added to other websites but no image for your newly created website “We.Train”.
3. To add small thumbnail image in front of your newly created website “We.Train”, perform following steps:

Click on website “We.Train”.

1. Click on “View Properties” button present at top.
2. Click on “Thumbnail” tab.
3. Click on “Upload Image” button. Select the image to upload.
4. Click on **Save**.

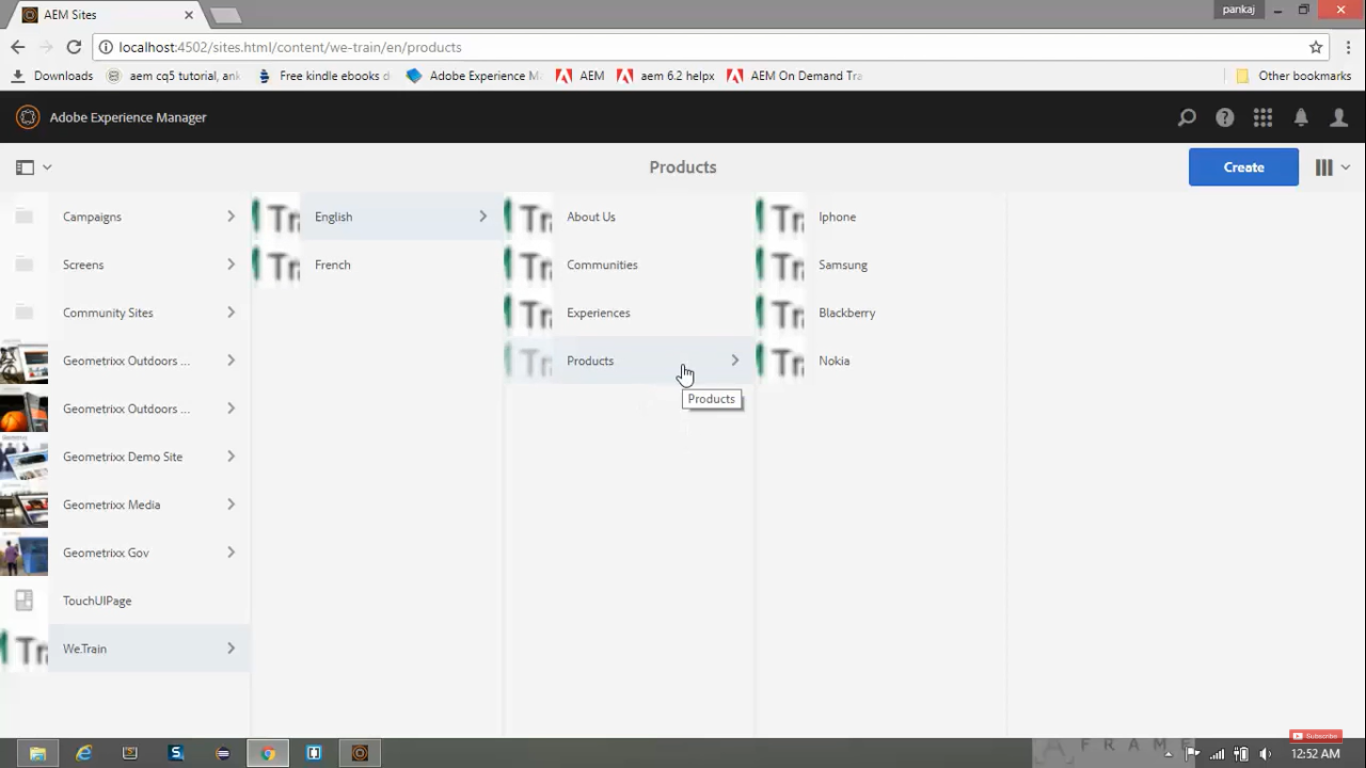






# 19. Create website structure in AEM:

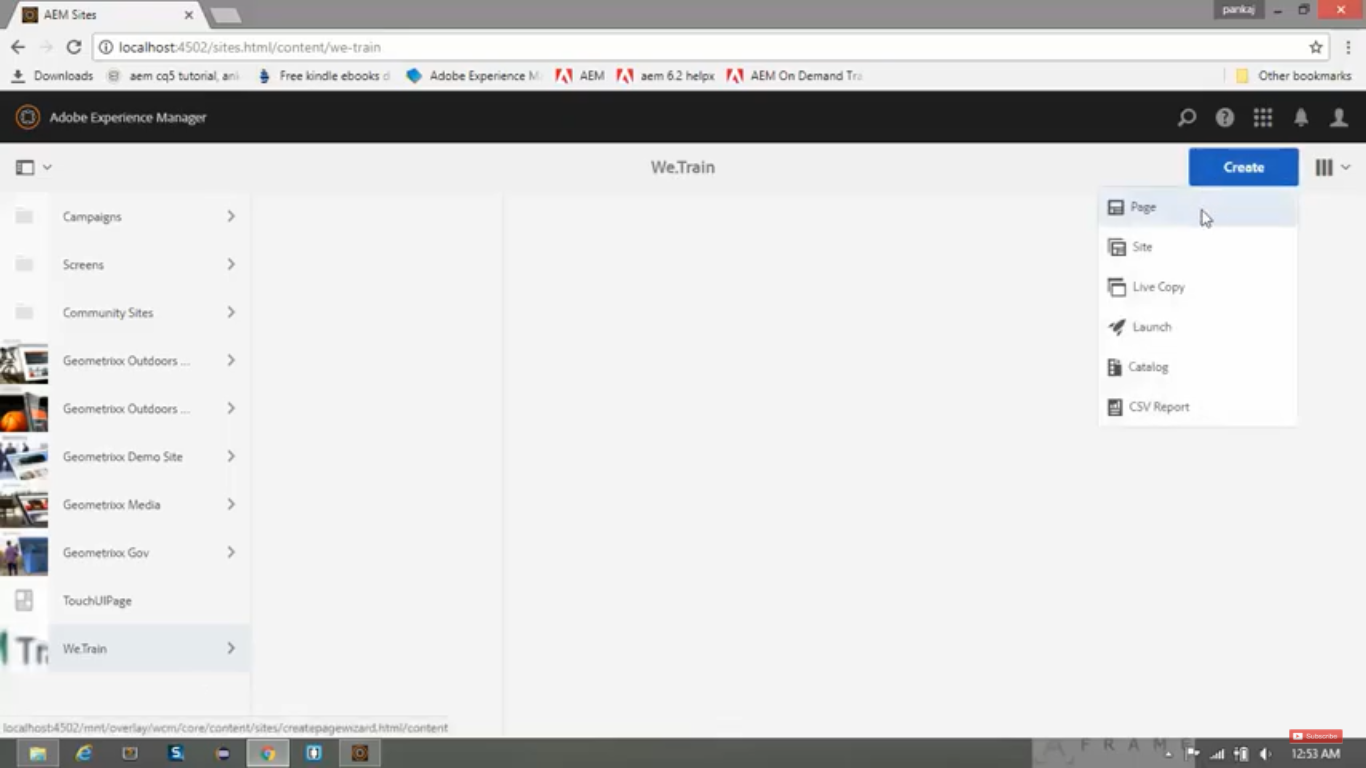
How to create below mentioned website structure ( image) to appear when we click on website “We.Train”

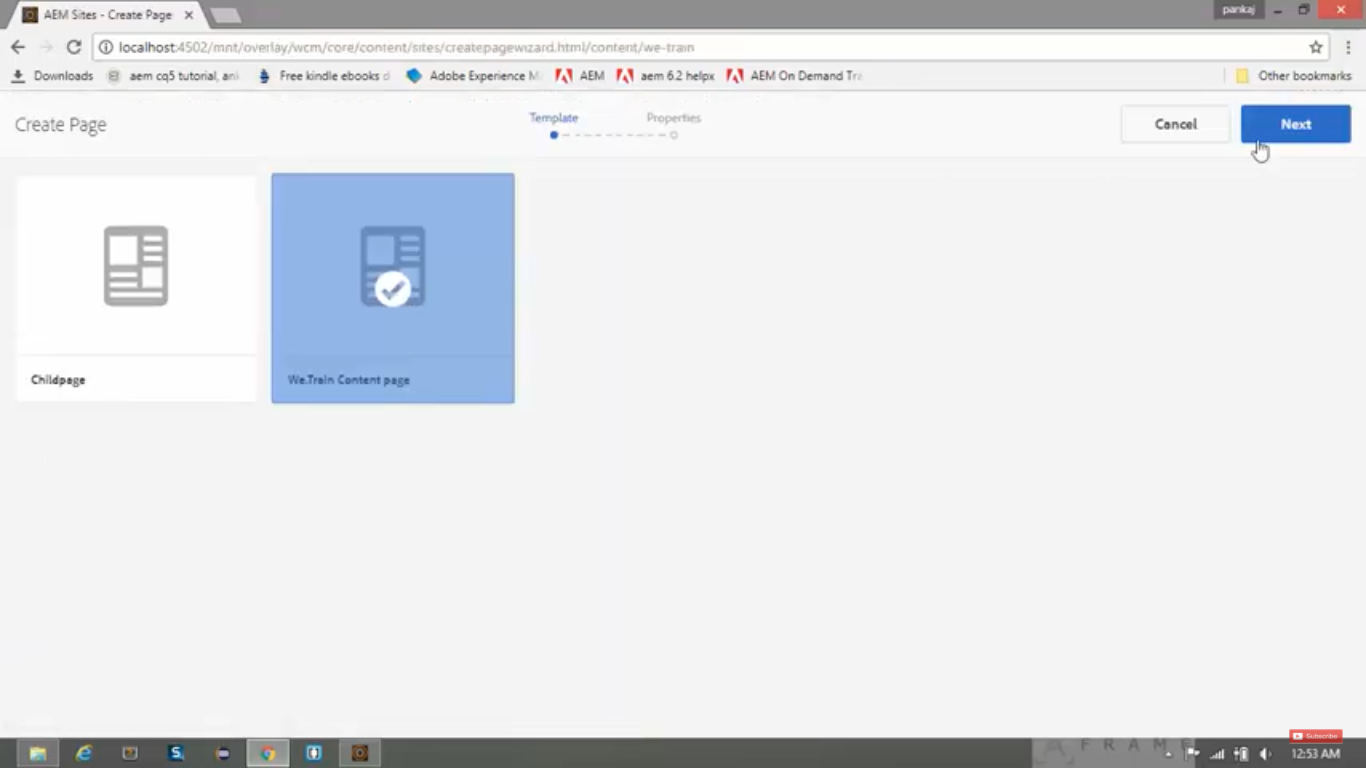


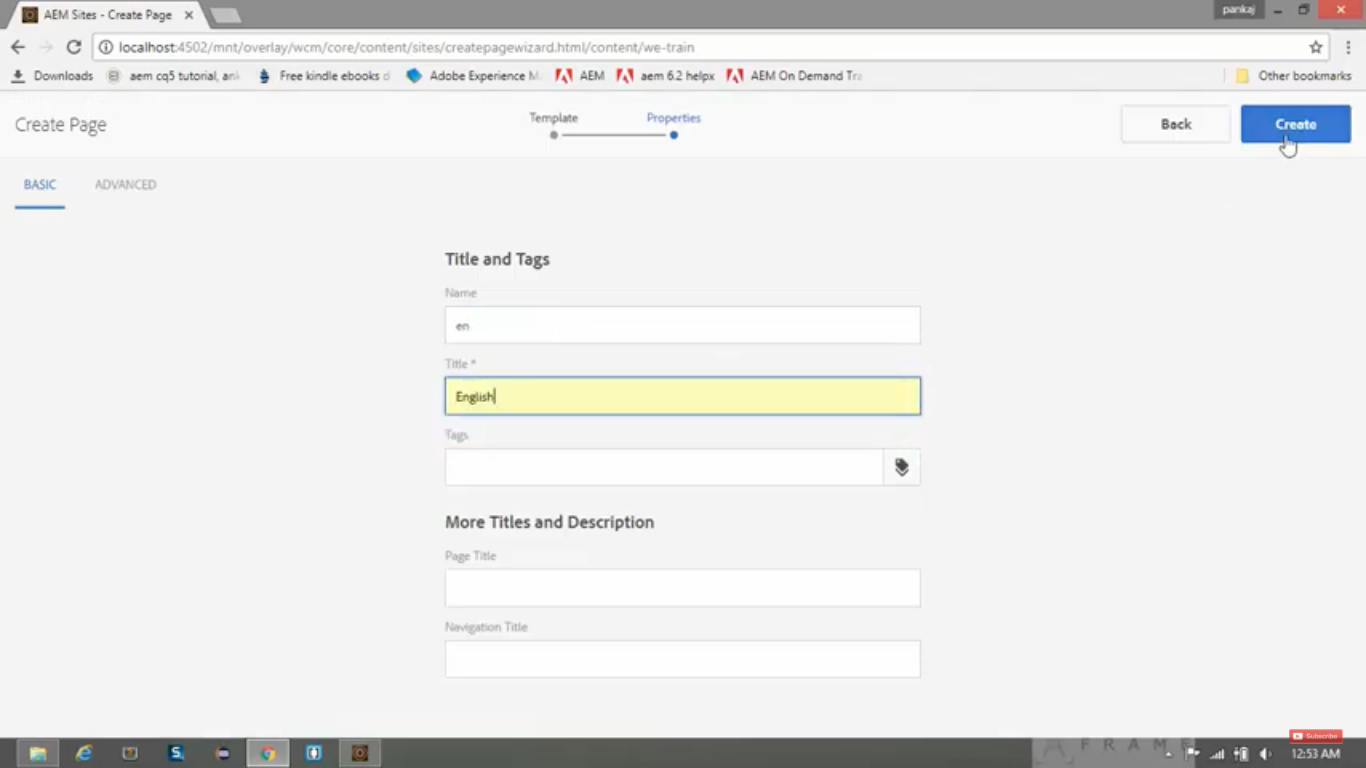
1. Go to **AEM --> Sites** --> Click on **We.Train** website --> Click on **Create** button --> Select **Page** option.
2. Select the “We.Train contents page” **Template** which we created.
3. Give values:

|  |  |
| --- | --- |
| **Field** | **Value** |
| Name | en |
| Title | English |

1. Click on **Create** button at top.
2. You can see the newly created page under **AEM --> Sites** --> We.Train --> English
3. Similarly create other pages.
4. See images on next page.







# 20. Sightly/HTL language:

1. **Sightly** is a *new templating language* developed by Adobe.
2. Currently it is called **HTL** (HTML Template Language) – *formerly known as* ***Sightly***.
3. It is **recommended to use Sightly** language **instead of JSP** while developing components in AEM.
4. **Slightly** to be **preferred** over **JSP** *while creating components* in AEM.
5. Sightly is **HTML5**.
6. A **Template** *created in* **Slightly** is a *valid HTML5 file*.
7. All ***HTL specific******syntax*** is **expressed within** the **data attribute** *within HTML tags*.
8. All **Slightly** specific **attributes** are *prefixed* with “**data-sly-**”.
9. **Sightly** is an *alternate* *of* **JSP**.
10. It takes the place of JSP as the *preferred* **Server-Side Template System** for HTML in AEM.

**Why do we need Sightly?**

1. Simplified Development IN AEM:

Because it allows HTML developers who have no knowledge of JAVA to participate more in AEM projects.

Whereas, in JSP, we write scriptlets in same file which look quite messy.

So, in comparison to JSP, Sightly is very simple and clean.

1. Security:

It is the *most important reason*.

Sightly ***automatically filters and escapes*** all text being output to presentation layer *to prevent from* **XSS attack** (Cross-Site Scripting Attack).

Whereas, In JSP, **developer** must ***manually ensure*** that **proper escaping is applied** *to each variable*.

But **Sightly** does it all **automatically**.

1. Separation of Concerns:

***Features*** in **Sightly** are purposely limited to ensure that to stays it **simple and straight-forward**.

All the **complex logic** should be kept in **external helper class** which our *HTL file can easily invoke*.

Example,

<p data-sly-use.obj=“script.js”>

$obj.text

</p>

Here, “**Data-sly-use**” is Sightly attribute.

We are invoking **script.js** *external file*. SO that ***complex logic*** *will sit inside* **script.js** file.

We are just invoking this complex logic (script.js) into our Sightly file.