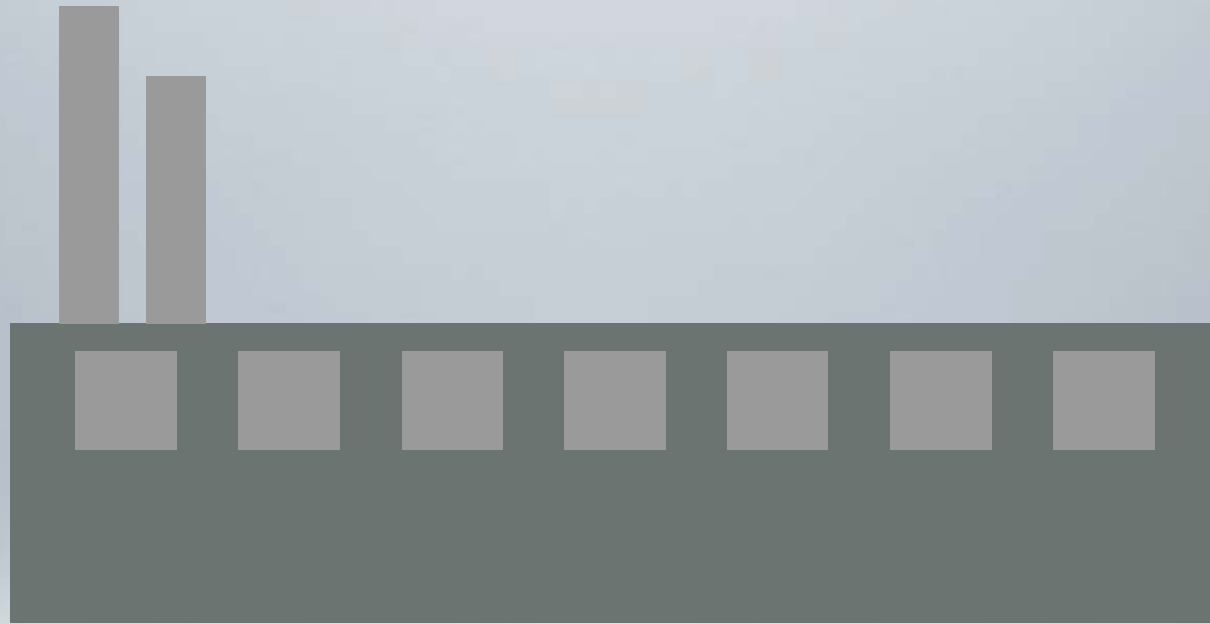


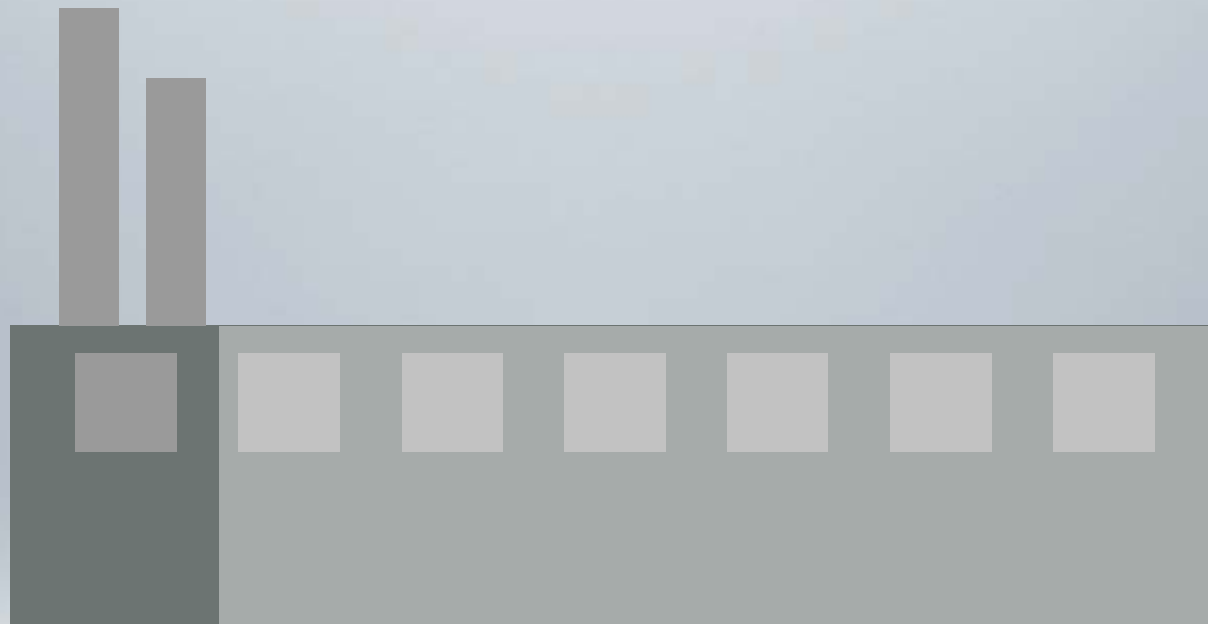
# WordPress as a Factory

Fundamentals of Plugin Development

# WordPress is like a Factory

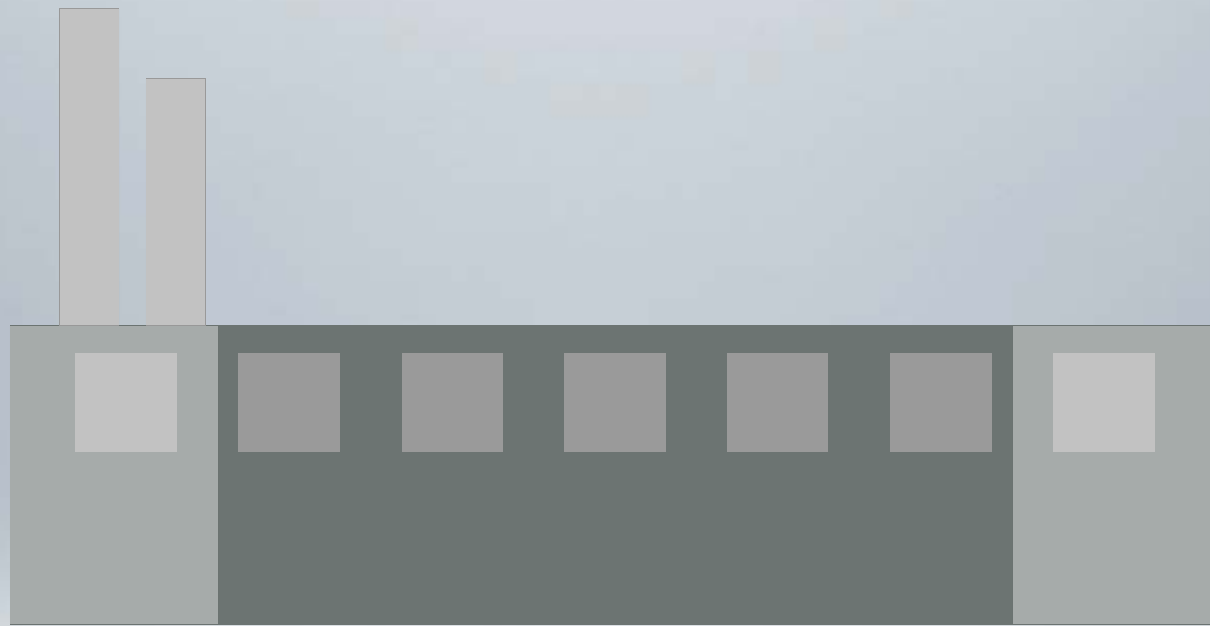


# WordPress is like a Factory



Sales  
Center

# WordPress is like a Factory



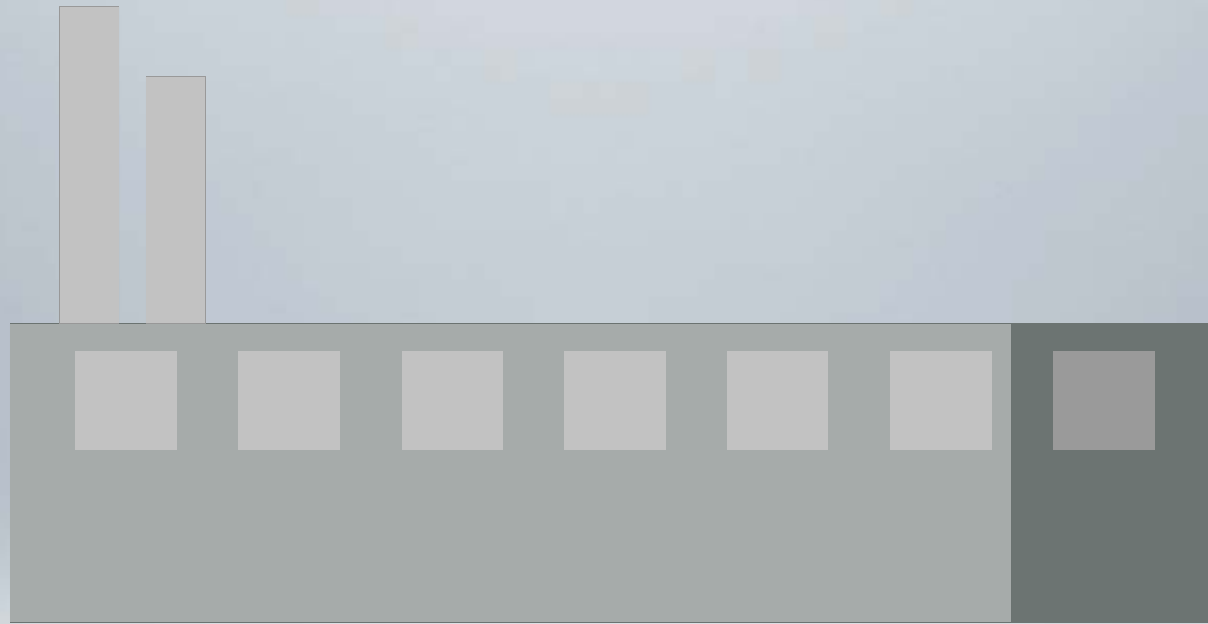
Production

# WordPress is like a Factory



Distribution

# The Distribution Center



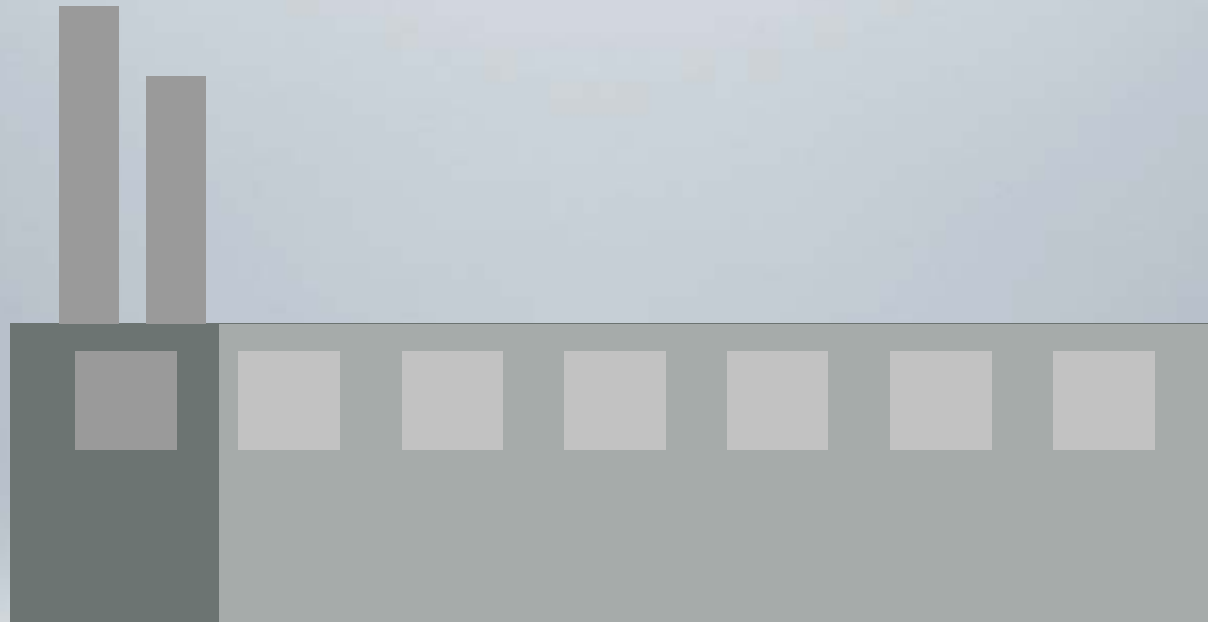
- This is how most people view WordPress
- The Distribution Center is where they go to pickup their product
- Depending on what happens at the sales floor, the product they pick up can be one of several different things.

# The Distribution Center



- Front page of your blog
- Login Page
- Admin Page
- RSS Feed
- 404s and PHP errors

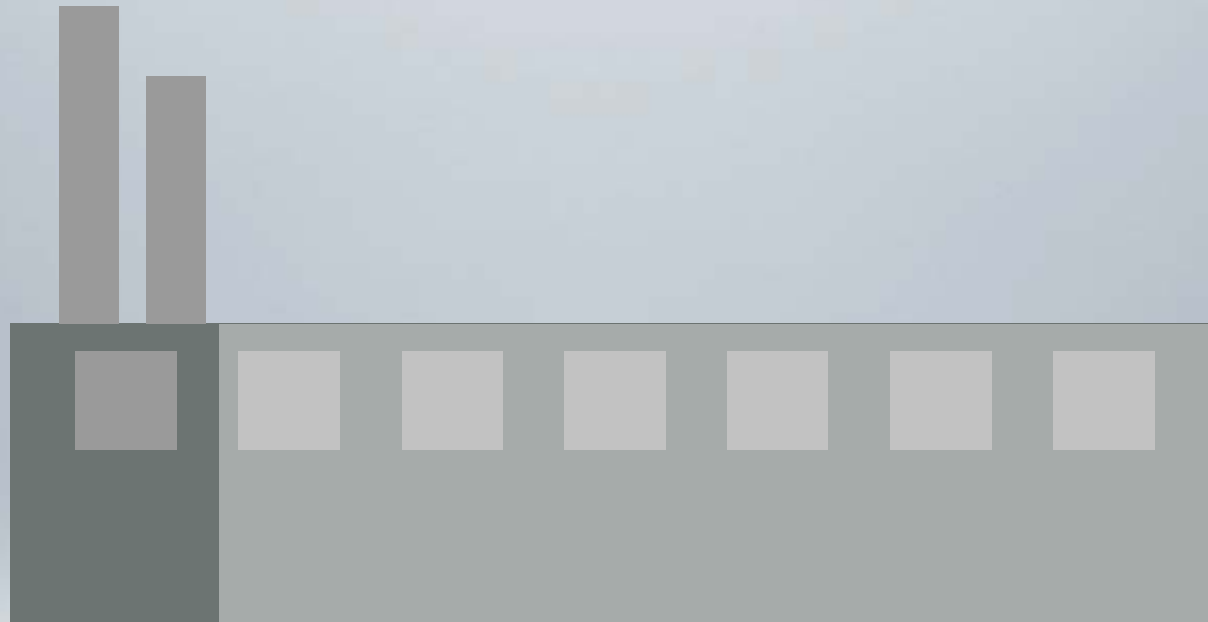
# The Sales Center



Everybody who has received a WordPress produced product has interacted with the sales center on one level or another

- Directly through their web browser
- Indirectly through an RSS client
- Maybe even the command line or a third party script

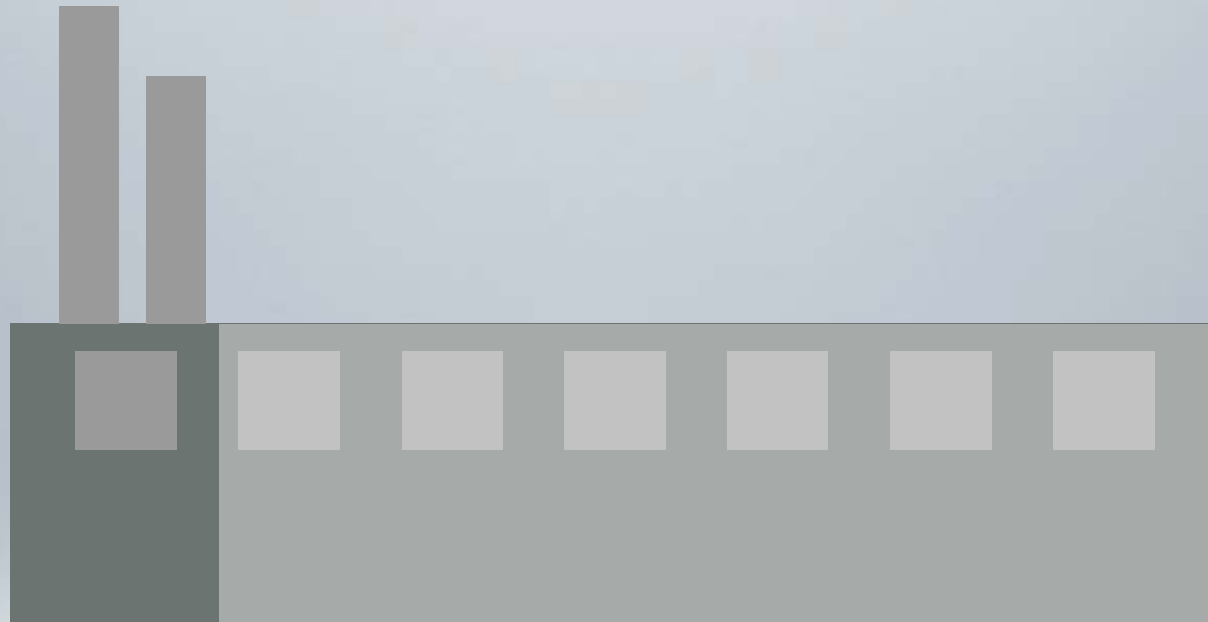
# The Sales Center



Sometimes you bring special requests with you:

- GET ( <http://example.com/?p=21> )
- POST ( Through a web form )
- Browser COOKIES

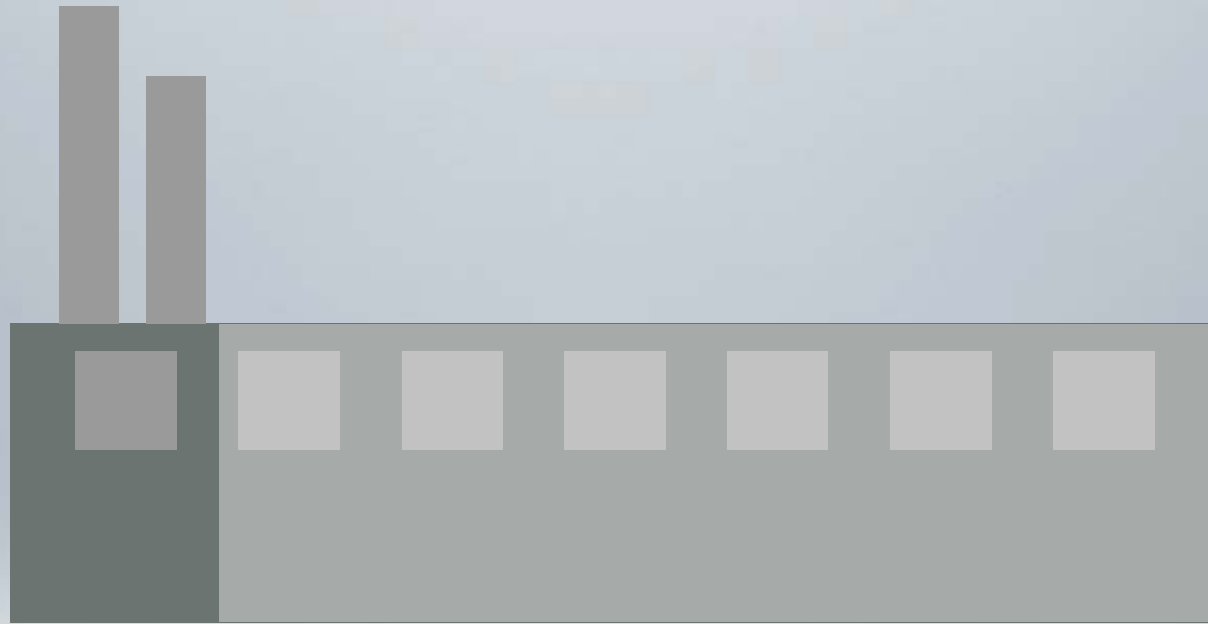
# The Sales Center



## Multiple Doors To Enter

- `index.php`
- `wp-admin/index.php`
- `wp-login.php`

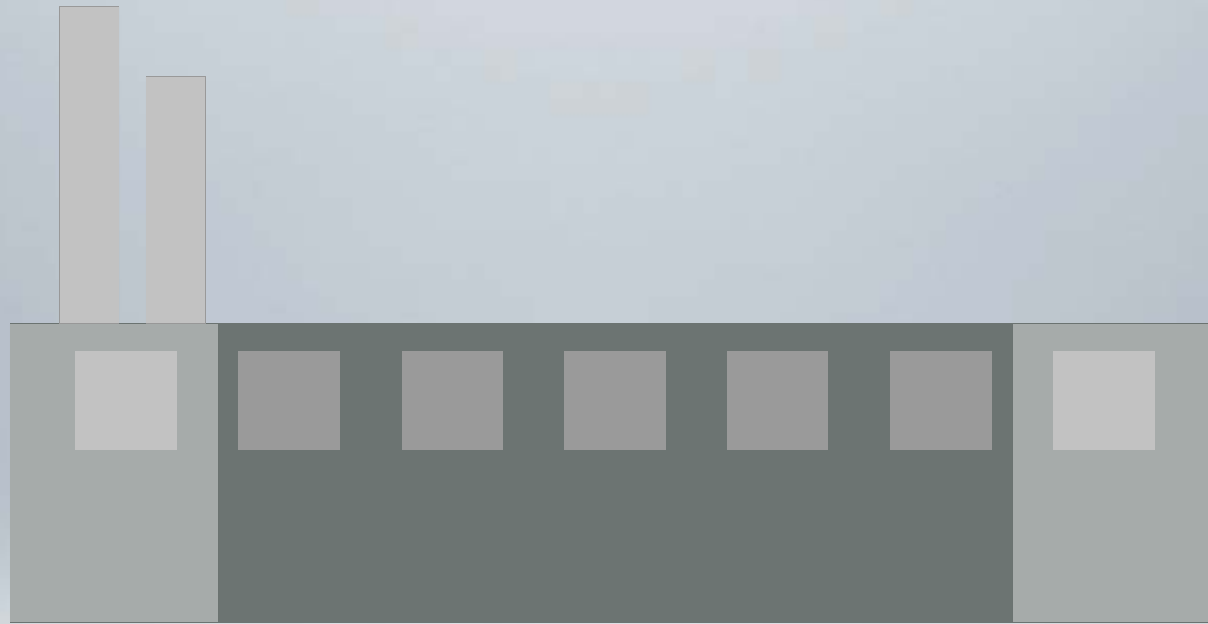
# The Sales Center



What happens in the sales center determines what can be displayed in the distribution center:

- How you enter the sales center ( Browser, RSS client, Google bot )
- What you bring with you ( GET, POST, COOKIE )
- Which door you choose ( index.php, wp-admin/index.php , etc )

# The Production Center



The key to developing efficient plugins is to thoroughly understand the Production Center

- The Product
- The Conveyor Belts
- The Machines

# The Product

How do we define our product?

- Our product is essentially, data stored in the server's memory
  - PHP Constants
  - Variables ( strings, integers, arrays, objects, etc )
  - COOKIES
  - SESSIONS

Where does the data come from before its placed in the above containers?

- Browser REQUESTS (GET and POST)
- PHP files
- Your Database (our warehouse)

# The Belt

The belt drives our product. It transports it from sales to distribution

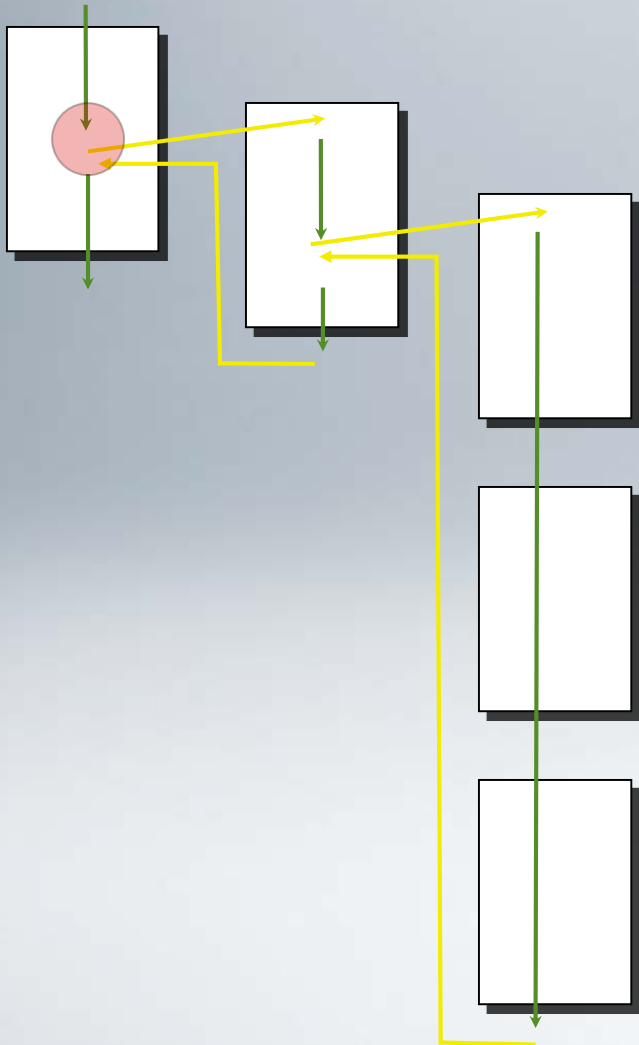
- What do we know about the belt?
  - It's built one section at a time
  - It can quickly and dynamically change directions based on our data
  - It is better represented as one large belt with loops rather than a tree with many branches. ( You could walk the belt from Distribution back to Sales )

That's cute, but how does it actually work?

- PHP includes() and the occasional function construct the belt one segment at a time based on the state of the data immediately before the belt segment is generated.
- Load the file, check it for syntax errors, process the file, and systematically follows includes() until the product is delivered or the belt brings you back to the same file.

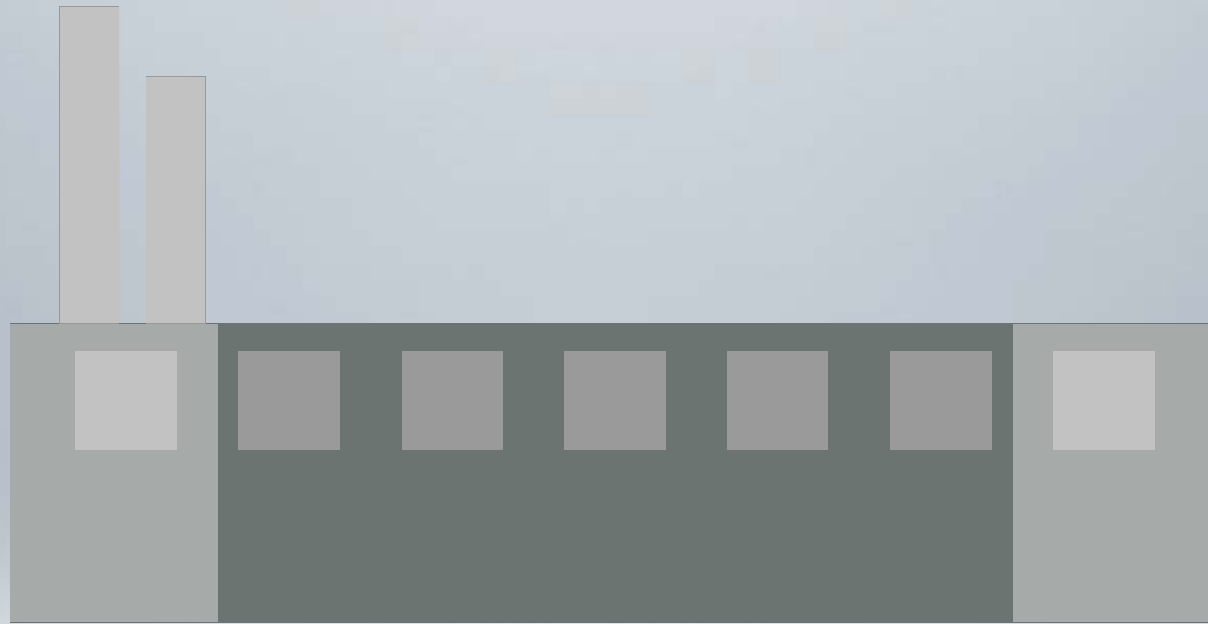
# The Belt

A visual representation of our “belt” or file includes()



- The first file is loaded, checked for errors, and processed
- It hits a PHP include, loads the requested file and processes down the script
- It in turn hits another include which processes down through two more includes
- At this point, the product has not be delivered (PHP is still processing data) and there are no includes at the bottom of this file
- So PHP returns to the next line in the file we left and continues to process to the end of that file
- At the end of file two, PHP returns to the next line in file one and continues to the end of the script.

# The Production Center



The key to developing efficient plugins is to thoroughly understand the Production Center

- The Product
- The Conveyor Belts
- The Machines

# Modifying the Production Center

What does it take to modify the production center properly and efficiently?

Good Timing

# Modifying the Production Center

What does it take to modify the production center properly and efficiently?

- What about actions and filters?
  - Actions and Filters are just special functions on the belt.
- There's only two of them.
  - `do_action()` and `apply_filters()`
- Both of those functions are concerned with the same array
  - All a plugin developer needs to expand the factory is timing.
  - You just have to know at what point in the belt would be the best time to do what you want to do
  - Once you've determined where you need to extend the functionality or at what point you want to alter the data, you only need to associate your custom function with that point on the belt

# Modifying the Production Center

So how do you add actions or filters?

- Find the `do_action()` or the `apply_filter()` you need along the belt (within the PHP files)
- Remember the name
- Use one of two functions in your plugin file
  - `add_action()` or `add_filter()`

# Modifying the Production Center

Using WordPress Hooks (`add_action` and `add_filter`)

- Remember that your plugin is included really early on in the belt
- You don't want to echo anything directly in your plugin file or you'll crash the belt
- Determine how many custom functions you'll need (how many places down the belt will you need to add some process or modify some data?)
- Create a function for each of those spots in the belt (and any helping functions you might need)
- Add each of them to an action or a filter depending on what you need to do.

# Modifying the Production Center

## Using WordPress Actions

- Create a plugin file and name it `no_access.php`
- Open that file up, put in the correct heading information and add the following function:
  - ```
function restrict_access_to_post(){  
    global $post;  
    if ( $post->ID == 2 ) {  
        die('Access Restricted');  
    }  
}
```
- Find out where the `$post` variable is first defined if you're looking at a single post.
- Find *any* `do_action()` reference after that but before your post is displayed, and hook your function to it:
  - ```
add_action( 'template_redirect' , 'restrict_access_to_post'  
);
```

# Modifying the Production Center

## Using WordPress Filters

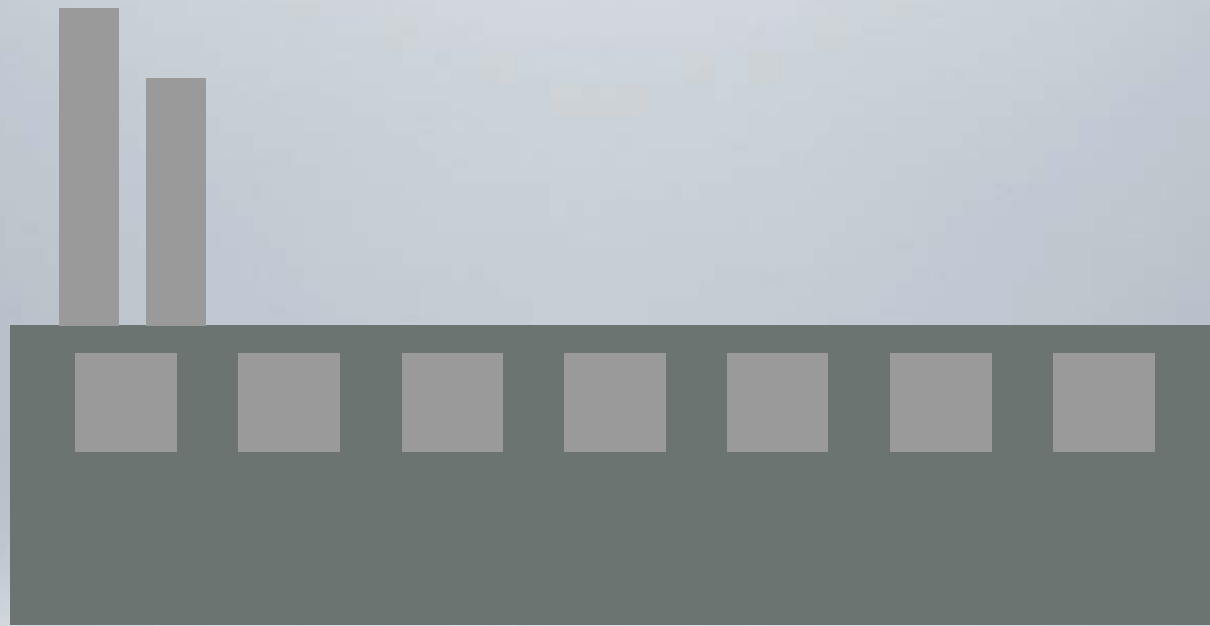
- Create a plugin file and name it `restrict_access_to_post.php`
- Open that file up, put in the correct heading information and add the following function:
  - ```
function restrict_access_to_post( $content ){  
    global $post;  
    if ( $post->ID == 2 ) {  
        return 'Access Restricted';  
    }  
}
```
- Find out where the post content is grabbed from the database and determine if a filter is available.
- If you determine that one is, hook your function to it:
  - ```
add_filter( 'the_content' , 'restrict_access_to_post' );
```

# Modifying the Production Center

## The difference between Actions and Filters

- Actions and Filters are both stored in the same array
  - `add_action( 'the_content' , 'my_function' );` would work just as well as `add_filter( 'the_content' , my_function' );`
- It's not the manner in which your functions are attached to `$wp_filter` array that makes actions and filters different.
- It's the way they are used by `do_action()` and `apply_filters()`
  - In short, `apply_filters` expects a value to be returned to it while `do_action` does not.

# WordPress is like a Factory

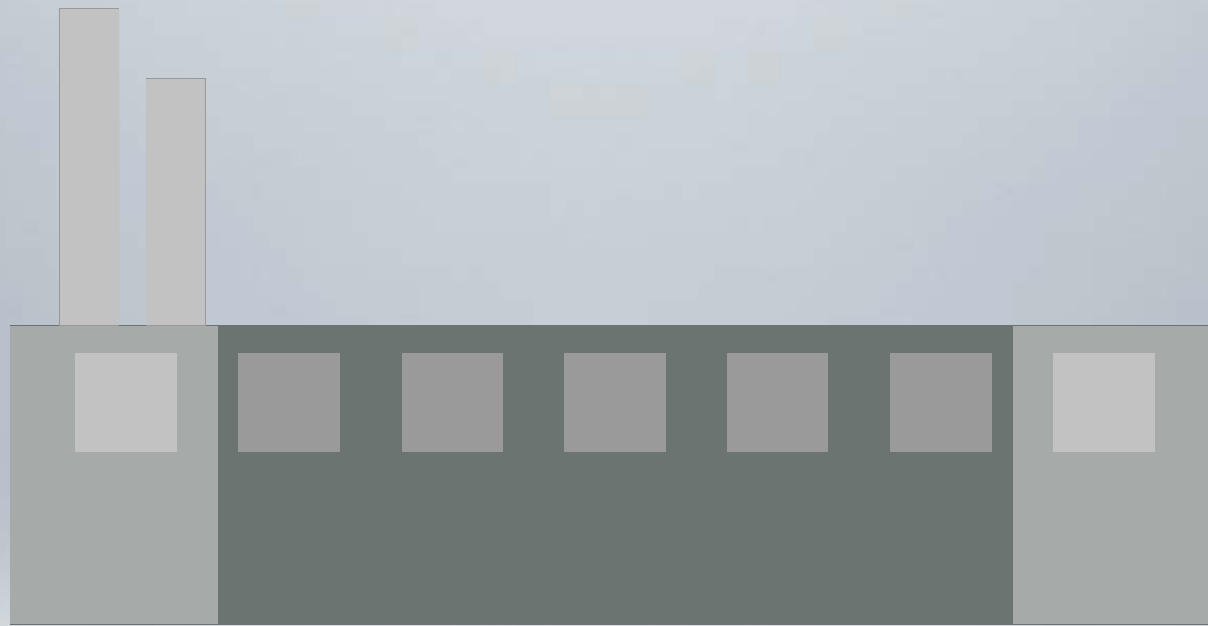


# WordPress is like a Factory



Sales  
Center

# WordPress is like a Factory



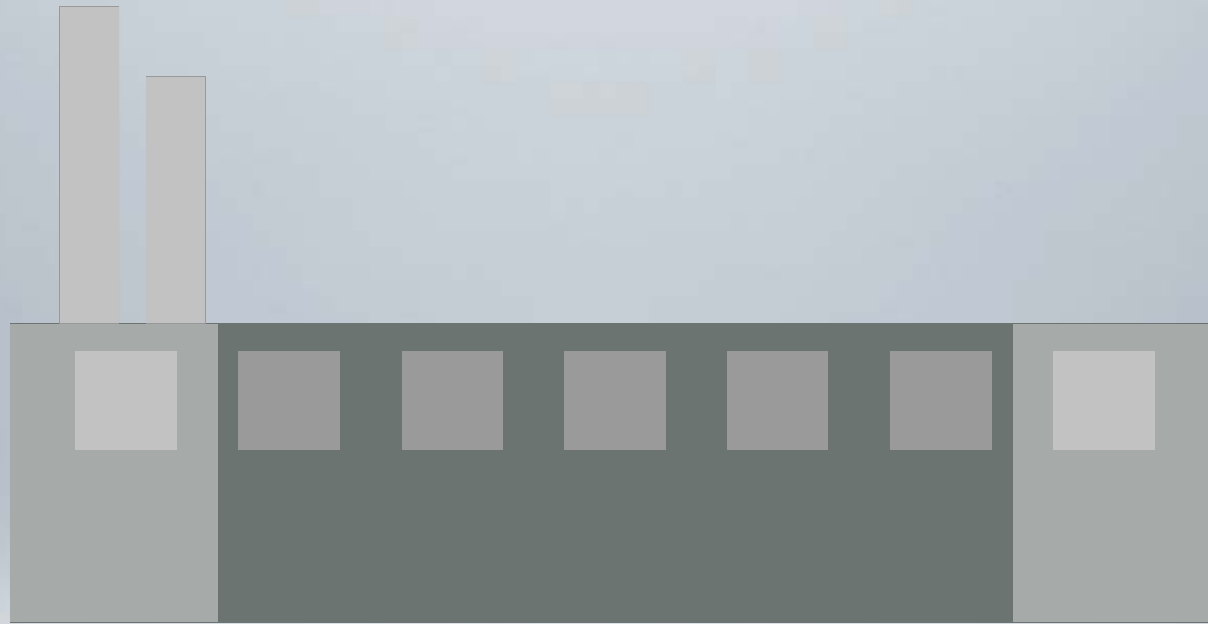
Production

# WordPress is like a Factory



Distribution

# WordPress is like a Factory



The key to becoming a successful plugin developer is understanding the production center. Once you understand the environment, what your factory can produce will only be limited by your PHP skills and your good or bad sense of timing.