**Pega Constellation 24.2 – Web Mashup Embedding Guide**

**📌 Overview**

In **Pega Constellation 24.2**, the **Web Embedding (Mashup)** feature allows you to embed Pega case types into an external web application using a <pega-embed> custom HTML element.  
This lets users interact with Pega cases directly inside your own website without navigating to Pega’s full portal.

**🛠 Prerequisites**

* **Pega Platform** 24.2 (Constellation UI)
* A **case type** (e.g., Credit Card Application)
* **VS Code** installed
* **Live Server** VS Code extension
* SSO authentication set up in Pega (if using Authorization Code grant type)

**📋 Step-by-Step Setup**

**1️⃣ Configure Web Embed Channel in Pega**

1. Go to **App Studio** → **Channels** → **Web Embed**.
2. Fill in:
   * **Channel name**: MashupTest
   * **Description**: MashupTest
   * **URL**: Your Pega environment URL (e.g., https://xxx.pega.net/prweb/)
   * **Grant Type**: Authorization code
   * **Authentication Service**: SSO (or as per your config)
   * **Action**: Create a case
   * **Case type**: Select your target case (e.g., *Credit Card Application*)
3. Optional:
   * Enable **Defer embed load** to load content only when required.
4. Save and click **Generate Web embed code**.

**Run via Local Server (Avoid Direct File Open)**

**Why?**

* Opening index.html directly via file:// will cause **CORS** or **server errors** because the browser blocks cross-origin requests without an HTTP(S) server.
* Running via http://localhost simulates a real web server environment.

**How to run:**

1. Open folder in **VS Code**.
2. Right-click index.html → **Open with Live Server**.
3. Browser opens at http://127.0.0.1:5500 (or similar).

**⚙ How It Works – Behind the Scenes**

**First — What is the DOM?**

* **DOM** = **Document Object Model**.
* When your browser loads a webpage, it takes the HTML and turns it into a **tree** of elements that JavaScript can see and interact with.
* Think of it like the blueprint of the page in memory:
  + <html> is the root
  + <head> and <body> are branches
  + Inside <body> you might have <div>, <p>, <pega-embed>, etc.
* JavaScript doesn’t edit “the HTML file on disk.”  
  It edits this **in-memory tree** — so when you say document.getElementById("theEmbed"), you’re asking the DOM:

“Hey blueprint, where’s that box with the ID ‘theEmbed’?”

**2️⃣ What happens when the browser loads your Pega mashup code**

Let’s use your **working example** and run it step-by-step:

1. **Browser starts reading HTML from top to bottom.**
2. It sees:

html

CopyEdit

<script src="https://wr4ihgbt.pegace.net/prweb/.../pega-embed.js"></script>

* + This **downloads the Pega Mashup JavaScript library**.
  + That file contains all the code that teaches the browser **what <pega-embed> even means**.  
    Without it, <pega-embed> is just an unknown tag.

1. Then the browser continues reading until it reaches:

html

CopyEdit

<pega-embed ...></pega-embed>

* + Now, because pega-embed.js was already loaded, the browser knows:

“Oh, this is a special custom element. When ready, it can fetch Pega content and display it.”

1. Then at the bottom, your code runs:

javascript

CopyEdit

document.addEventListener("DOMContentLoaded", function () {

var embed = document.getElementById("theEmbed");

if (embed && typeof embed.load === "function") {

embed.load();

}

});

* + **DOMContentLoaded** means: wait until the **DOM** (that blueprint) is **fully built**.
  + Then find <pega-embed id="theEmbed"> in that blueprint.
  + Then check if it has a .load() function (added by pega-embed.js).
  + If yes, call .load() → this triggers the **fetch** from Pega’s server.
  + That fetch goes to https://wr4ihgbt.pegace.net/prweb/ with the case info, auth, etc.
  + The returned HTML/UI from Pega is inserted into the shadow DOM inside <pega-embed> and you see the case UI.

**3️⃣ Why the short non-working version stays blank**

In your short code:

html

CopyEdit

<script src="...pega-embed.js"></script>

<pega-embed ... deferLoad="true"></pega-embed>

* You told Pega:

“Don’t load right away” (deferLoad="true").

* But you never gave the .load() command — so nothing ever happens.
* Even if you removed deferLoad="true", it might still fail if:
  + The <pega-embed> appears before pega-embed.js is fully loaded.
  + The element doesn’t have height, so it’s invisible.

**4️⃣ Key takeaway**

* **The DOM** is the browser’s in-memory representation of the page.
* **DOMContentLoaded** makes sure the whole page structure exists before we try to manipulate it.
* **pega-embed.js** is what teaches the browser how to make <pega-embed> actually load Pega content.
* **deferLoad="true"** means you must manually call .load() — otherwise it’s an empty shell forever.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.