**✅ Getting Started with the Error Handling Package in Automation Anywhere A360**

**📌 What is the Error Handling Package?**

The **Error Handling package** in Automation Anywhere A360 provides actions to **catch, handle, and manage errors** (also called exceptions) during bot execution. It helps you build **robust and fault-tolerant bots**.

**🎯 Why Use Error Handling?**

* Prevent bot from stopping on errors
* Log errors for debugging
* Take alternative actions (fallback steps)
* Improve reliability in production

**🔧 Key Actions in the Error Handling Package**

| **Action** | **Description** |
| --- | --- |
| **Begin Error Handling (Try)** | Starts the try block where code is monitored for exceptions |
| **Set Catch** | Defines what to do if an error occurs in the Try block |
| **Set Finally** | Defines what to do after Try/Catch (always executes) |
| **Throw Error** | Manually raises a custom error |
| **Get Error** | Gets the error message, line number, error code, etc., from the caught error |

**⚙️ How to Use Error Handling – Example**

**Scenario: You are opening an Excel file that may or may not exist.**

**Step 1: Add Begin Error Handling (Try)**

* Inside this, add action to **open Excel file** using the Excel package.

**Step 2: Add Set Catch**

* Add **Log to File** or **Send Email** action to inform about the error.
* Use $ErrorMessage$, $ErrorLineNumber$, etc., from **Get Error**.

**Step 3: Add Set Finally**

* Close the Excel file or clean up temporary variables.

**✅ Example Bot Flow:**

plaintext

CopyEdit

Begin Error Handling

→ Open Excel File (path: $FilePath$)

Set Catch

→ Get Error → Save to variable: $errorMessage$

→ Log to File: "Error occurred: $errorMessage$"

Set Finally

→ Close Excel

**🧠 Error Handling Variables**

| **Variable** | **Description** |
| --- | --- |
| $ErrorMessage$ | The error message text |
| $ErrorLineNumber$ | The step number where the error occurred |
| $ErrorCode$ | The code associated with the error |
| $ErrorSource$ | The action name that caused the error |

**📘 Best Practices**

* Always use **Try-Catch** around:
  + File operations (open/read/write)
  + Excel, Database, API, Web automation
* Use **Log to File** in Catch to record errors
* Use **Finally** to clean resources (close files, disconnect DB)
* Avoid overusing Try-Catch for every small action (group them logically)

**🎯 Interview Questions & Answers – Error Handling in Automation Anywhere**

**Q1. What is the purpose of the Error Handling package in A360?**

**Answer:**  
It allows bots to catch and manage runtime errors using structured blocks like Try, Catch, Finally, preventing bots from failing unexpectedly.

**Q2. What are the main actions in Error Handling?**

**Answer:**

1. Begin Error Handling (Try)
2. Set Catch
3. Set Finally
4. Throw Error
5. Get Error

**Q3. What is the difference between Catch and Finally?**

**Answer:**

* **Catch**: Executes only **if an error occurs** inside the Try block.
* **Finally**: Executes **always**, regardless of whether an error occurred or not.

**Q4. How can you get the details of an error?**

**Answer:**  
Use the **Get Error** action inside the Catch block to retrieve variables like:

* $ErrorMessage$
* $ErrorLineNumber$
* $ErrorCode$
* $ErrorSource$

**Q5. Can we throw a custom error manually?**

**Answer:**  
Yes, using the **Throw Error** action. You can define a custom error message which will trigger the Catch block.

**Q6. What are some real-life use cases for Try-Catch in bots?**

**Answer:**

* Opening a file that may be missing
* Logging in to a website with unreliable network
* Reading from an Excel cell that might be empty
* Sending an email when a transaction fails

**Q7. What happens if you don’t use Try-Catch and an error occurs?**

**Answer:**  
The bot will **stop immediately** and the error will be shown in the Control Room logs. There will be **no opportunity to recover** or log the issue gracefully.

**Q8. Can you nest Try-Catch blocks inside one another?**

**Answer:**  
Yes, A360 allows **nested error handling** for complex scenarios, but it should be used carefully to avoid confusion.